Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1995-06

Stability analysis of a 2-D acoustic/structure model

Shehan, Joe Michael.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/31489

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS
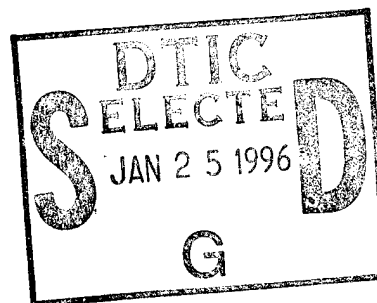
STABILITY ANALYSIS OF A 2-D
ACOUSTIC/STRUCTURE MODEL

by

Joe Michael Shehan

June 1995

Thesis Advisor: Fariba Fakhroo

DTIC QUALITY INSPECTED 3

1996 0118 023

DTIC
SELECTED
JAN 2 5 1996
G

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE June 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE STABILITY ANALYSIS OF A 2-D ACOUSTIC/STRUCTURE MODEL	5. FUNDING NUMBERS
6. AUTHORS Shehan, Joe M.	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT(*maximum 200 words*)

Reliably modeling noise attenuation through interaction with vibrating boundary structures is fundamental to the formulation of effective active noise control systems. In this paper we investigate, through numerical approximation, uniform exponential stability of two systems which model the acoustic/structure interaction of an air-filled, rectangular cavity. The first model assumes dissipative boundary conditions along one side of the boundary, while the second assumes dissipative boundary conditions along all four sides of the cavity. We obtain weak variational formulations for these models, express each as finite dimensional systems, and use the Galerkin technique to transform the distributed parameter systems into systems of ordinary differential equations. We analyze the stability of the finite dimensional systems in order to gain insight into the stability of the original infinite dimensional systems. Essentially, our analysis consists of solving a generalized eigenvalue problem and observing where the eigenvalues lie within the complex plane. This stability analysis leads us to conclude that one model is better suited for use in the formulation of the noise control problem.

14. SUBJECT TERMS Distributed Parameter Systems, Galerkin Method, Fluid/Structure Interaction, Uniform Exponential Stability, General Eigenvalue Problem	15. NUMBER OF PAGES 90
	16. PRICE CODE

17. SECURITY CLASSIFI- CATION OF REPORT Unclassified	18. SECURITY CLASSIFI- CATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI- CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

STABILITY ANALYSIS OF A 2-D ACOUSTIC/STRUCTURE MODEL

Joe Michael Shehan
Major, United States Marine Corps
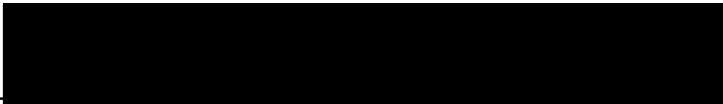B.S. in Mathematics, University of North Carolina at Greensboro, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS
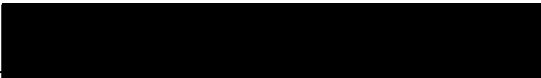
from the

**NAVAL POSTGRADUATE SCHOOL
June 1995**
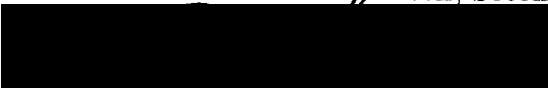
Author: _____
Joe Michael Shehan

Approved by: _____
Fariba Fakhroo, Thesis Advisor

_____
Van Emden Henson, Second Reader

_____
Richard Franke, Chairman, Department of Mathematics

ABSTRACT

Reliably modeling noise attenuation through interaction with vibrating boundary structures is fundamental to the formulation of effective active noise control systems. In this paper we investigate, through numerical approximation, uniform exponential stability of two systems which model the acoustic/structure interaction of an air-filled, rectangular cavity. The first model assumes dissipative boundary conditions along one side of the boundary, while the second assumes dissipative boundary conditions along all four sides of the cavity. We obtain weak variational formulations for these models, express each as finite dimensional systems, and use the Galerkin technique to transform the distributed parameter systems into systems of ordinary differential equations. We analyze the stability of the finite dimensional systems in order to gain insight into the stability of the original infinite dimensional systems. Essentially, our analysis consists of solving a generalized eigenvalue problem and observing where the eigenvalues lie within the complex plane. This stability analysis leads us to conclude that one model is better suited for use in the formulation of the noise control problem.

v

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Imagine that you are ten years old and your teenage brother asks you to participate in a scientific experiment. You agree. Inside a 55 gallon drum you go. Your brother, clever lad that he is, manages to suspend the drum (and you) a few feet in the air. So there you are, dangling by a rope three feet off the ground–sealed in a metal drum. With *your* baseball bat in hand, he strikes the drum dead center. After the drum stops reverberating, he lowers it to the ground. As he pops the lid off, he asks you to describe how the acoustic noise field in the drum behaved after his forceful swing. As a composed budding young scientist, you disregard the blood dripping from your left earlobe and answer, "Initially, a large noise field was created by the strike of the bat. However, after the impulse force was experienced, the intensity of the noise field steadily decreased until it eventually was imperceptible." Your brother thanks you dearly and then proposes a heat conduction experiment...

In this paper, we too examine noise transmission attenuation through a vibrating boundary structure. Our approach, however, is to do so through numerical approximation. More specifically, we examine the exponential stability of infinite dimensional second order systems of coupled partial differential equations by numerical approximation. This is a topic attracting considerable interest in the fields of engineering and applied mathematics because the applications are both numerous and diverse—reducing noise levels in automobiles, aircraft, and space launch vehicles to name a few. The degree to which a control system formulated to effect noise reduction in a fluid-filled cavity succeeds depends to great extent on how accurately the underlying mathematical model agrees with the observed physical phenomena. As our young scientist reported above, the acoustic field created by an external force acting on the cavity boundary steadily diminished to zero as time passed (in the absence of any sustaining force). This is equivalent to requiring that all solutions to the system of equations selected to model the behavior of the acoustic field, as well

1

as the vibrating boundary, converge exponentially to zero by a uniform rate of decay. Establishing this result for the infinite dimensional system is nontrivial and, at times, a very difficult task.

To help clarify these ideas, consider the following abstract formulation. According to [Ref. 1], many examples related to acoustics or fluid/structure interactions can be modeled abstractly as a first order system of equations as follows:

$$y_t(t) = Ay(t), \quad t > 0, \quad y(t) \in \mathcal{H}, \tag{I.1}$$

where $\mathcal{H}$ is an appropriately defined Hilbert space. The companion linear control system is typically written

$$y_t(t) = Ay(t) + Bh(t), \quad h(t) \in \mathcal{R}^n, \tag{I.2}$$

where h is a control input and B is a linear operator from $\mathcal{R}^n$ into $\mathcal{H}$. This system possesses uniform exponential stability if there exist $M > 0$ and $\beta > 0$ such that for all $t \geq 0$ and for all $(y(0), y_t(0)) \in \mathcal{H}$

$$\|y(t), y_t(t)\|_{\mathcal{H}} \leq Me^{-\beta t}\|(y(0), y_t(0))\|_{\mathcal{H}} \quad [Ref.2, 3],$$

where $\|y(t), y_t(t)\|_{\mathcal{H}}$ denotes the energy of the system at time t and $\|(y(0), y_t(0)\|_{\mathcal{H}}$ denotes the energy of the system at $t = 0$.

The authors of [Ref. 1] state that "the most common approach for the approximation of a control problem involving I.2 is to formulate a sequence of finite dimensional control systems of the form

$$y_t^N(t) = A^N y^N(t) + B^N h(t), \quad t > 0, \quad y^N(t) \in \mathcal{H}^N, \tag{I.3}$$

where the dimension of the finite solution space $\mathcal{H}^N$ increases toward infinity as N tends to infinity. In general, equation I.3 is derived from I.2 using space discretization techniques such as finite difference, finite elements or spectral methods developed for the approximation of the solutions of I.1. A control strategy is then designed for the finite dimensional control problem involving I.3. This control is used as an

approximation to the desired control function for the infinite dimensional control problem I.2. One of the most practical conditions to assure the well-posedness of the finite dimensional control problem, as well as the convergence of the approximate controls to the desired control for the infinite-dimensional system, is that the solutions of I.3 for $h \equiv 0$ preserve the exponential decay of the solutions of I.1." Hence determining the stability of the finite dimensional system

$$y^N(t) = A^N y^N(t), \quad t > 0, \quad y^N(t) \in \mathcal{H}^N \tag{I.4}$$

is of fundamental importance.

Typically this stability analysis is accomplished by examining where the eigenvalues of A^N lie within the complex plane, since theory tells us that system I.4 is globally stable if and only if all eigenvalues of A^N have negative real parts [Ref. 4]. The process is nontrivial because numerical results have indicated that many popular approximation schemes fail to maintain a uniform decay rate as the dimension of the approximating system I.4 increases [Ref. 1], even in cases where the original system was proven to be exponentially stable.

In this paper, we examine two different infinite dimensional models by numerical approximation in order to gain insight into their adequacy for control system formulations. Model I is believed to be stable, but not uniformly exponentially stable [Ref. 5]. Model II is believed to be exponentially stable [Ref. 6].

In the following chapters, topics introduced above are addressed in greater detail. In Chapter II, we develop a general and two specific models describing an acoustic field inside a two-dimensional fluid-filled cavity surrounded by a perturbable boundary. In Chapter III, we illustrate the Galerkin technique chosen for our numerical approximations, and in Chapter IV, we apply this technique to the models under consideration. In Chapter V, we describe how specific approximations were obtained and present our results. We conclude with summary comments and propose future areas of study in Chapter VI.

II.   MATHEMATICAL MODELS
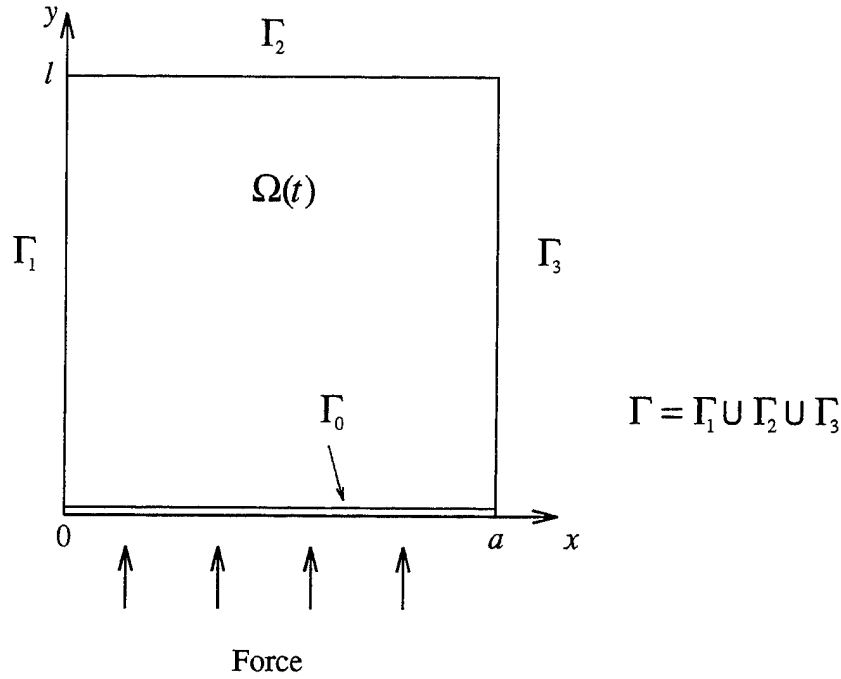
Figure 1. 2-D Acoustic Chamber for the General Mathematical Model

In this chapter, we develop a general and two specific models for a system consisting of the wave equation coupled with the beam equation on part of the boundary. Consider the two-dimensional rectangular air filled cavity surrounded by an impenetrable boundary shown in Figure 1. A noise source exterior to the cavity produces a perturbing force f which induces vibrations in the cavity boundary causing fluctuations (i.e., undesirable noise) in the acoustic pressure field within the cavity. Equations of motion describing boundary vibrations and acoustic pressure fluctuations within the cavity, together with appropriate initial value and boundary conditions form a system of coupled, second order partial differential equations in time.

We begin formulation of the general model by assuming that the interior acoustic pressure field satisfies the standard wave equation $\phi_{tt} = c^2 \Delta \phi$ where ϕ is the velocity potential throughout the cavity and c is the uniform speed of sound in the

fluid. The velocity potential ϕ is a complex-valued function satisfying

$$\vec{v}(t, x, y) = -\nabla \phi(t, x, y) \,,$$

where $\vec{v}$ denotes the fluid velocity. Initial value conditions $\phi(0, x, y)$ and $\phi_t(0, x, y)$, as well as boundary conditions along $\partial\Omega$—either Dirichlet, Neumann or dissipative—are specified.

A Newtonian analysis of forces and bending moments leads to equations describing the motion of the elastic walls bounding the cavity. For simplicity we assume that the boundary walls are impenetrable and that only one side of the rectangular boundary is perturbable (The term beam refers to the perturbable boundary.). We assume an *Euler-Bernoulli beam* where (i) $w(t, x)$ denotes the transverse displacement of the beam of length a, (ii) ρ_b and ρ_f denote the uniform mass densities of the beam and fluid, respectively, (iii) $M(t, x)$ denotes the total internal moment of the beam, and (iv) $f(t, x)$ denotes the external forcing term. The beam equation takes the general form

$$\rho_b w_{tt}(t, x) + M_{xx}(t, x) = -\rho_f \phi_t(t, x, w(t, x)) + f(t, x) \,, \qquad \text{(II.1)}$$

where $-\rho_f \phi_t(t, x, w(t, x))$ is the acoustic pressure (This is the first coupling term we see in the acoustic/structure system.).

Initial value conditions are specified, and boundary conditions for the beam indicate whether the ends are free, partially restrained, or clamped. Additionally, assumptions specifying the types of internal moments of the beam are necessary. Internal moments typically consist of bending and damping moments,

$$M(t, x) = M_{bending} + M_{damping} \,,$$

where

$$M_{bending} = \text{strain component} = E(x)I(x)w_{xx}(t, x) \,.$$

The stiffness of the beam is given by $E(x)I(x)$, where $E(x)$ is the Young's modulus and $I(x)$ is the cross-sectional area of the beam. $M_{damping}$ is taken to be either

6

Kelvin-Voigt, spatial hysteresis, or time hysteresis damping. While spatial hysteresis damping is an appropriate choice for a beam constructed of composite materials and time hysteresis damping is an appropriate choice for a beam constructed of "material with memory", Kelvin-Voigt damping assumes a memoryless beam of uniform (linear) mass density where the damping stress is proportional to the strain rate. That is,

Kelvin-Voigt damping: $M_{damping}$ = strain rate component = $c_D(x)I(x)w_{xxt}$,

where $c_D I(x)$ is the product of the Kelvin-Voigt damping coefficient $c_D(x)$ and beam cross-sectional area $I(x)$ [Ref. 7]. Here we consider a beam of uniform cross sectional area and density. Thus we assume Kelvin-Voigt damping and take the coefficient functions $c_D(x)I(x)$ and $E(x)I(x)$ to be constant for all x (i.e., $c_D(x)I(x) \longrightarrow c_D I$ and $E(x)I(x) \longrightarrow EI$).

Based upon the above discussion, the generalized mathematical model for the acoustic/structure system is:

$$
\begin{aligned}
\phi_{tt} &= c^2 \Delta \phi \quad \text{for } (x,y) \in \Omega , \ t > 0, \\
\rho_b w_{tt}(t,x) &+ M_{xx}(t,x) = -\rho_f \phi_t(t,x,w(t,x)) + f(t,x), \ 0 < x < a, \ t > 0,
\end{aligned}
\tag{II.2}
$$

with appropriate initial value and boundary conditions specified for the wave and beam equations.

Next we present two specific models and formally show that each model can be expressed in both weak and strong formulations—which, as we shall see, are equivalent given appropriate choices of inner product spaces. This preliminary work will formally justify expressing Models I and II as first order systems in time thus facilitating our numerical examinations of solution stability. In Chapter III we examine in some detail the popular variational scheme, the Galerkin method, used to obtain approximate solutions to coupled systems such as those addressed in this paper. We use the Galerkin scheme to transform the infinite dimensional system II.2 into a finite dimensional system to facilitate numerical analysis.

A. MATHEMATICAL MODEL I
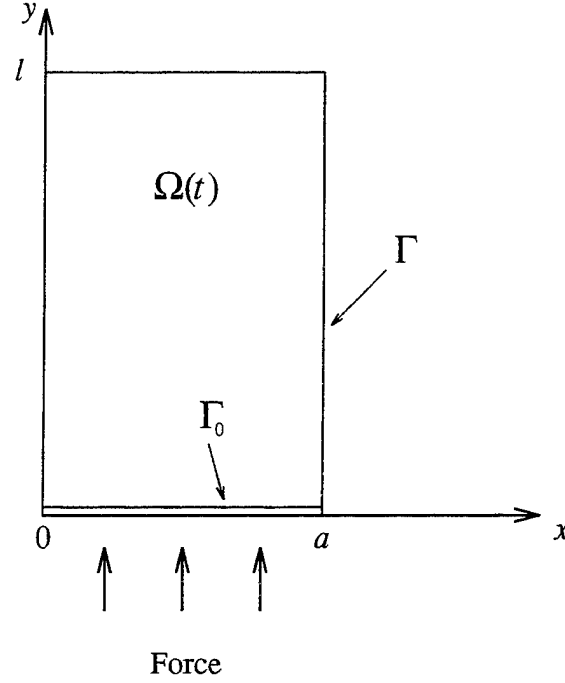
Figure 2. 2-D Acoustic Chamber for Model I

The theoretical model considered in this section is shown in Figure 2. Here the wave equation is coupled with the beam equation on one side of a 2-D rectangular, air filled cavity. We assume Neumann boundary conditions along Γ. That is,

$$\nabla \phi \cdot \hat{n} = 0, \quad (x,y) \in \Gamma, \quad t > 0,$$

where $\hat{n}$ represents the the outer normal with respect to Γ. The boundary is, in effect, a sound-insulated rigid wall which prevents any acoustic energy from escaping, or alternatively, the boundary is a perfect reflector of acoustic waves.

Further, we assume that the perturbable boundary (i.e., beam) can be characterized as an impenetrable fixed-end Euler-Bernoulli beam with Kelvin-Voigt damping and that both ends of the beam are clamped. Given these assumptions, the equations of motion describing the vibrations of the perturbable boundary are:

$$\rho_b w_{tt}(t,x) + M_{xx}(t,x) = -\rho_f \phi_t(t,x,w(t,x)) + f(t,x), \quad 0 < x < a, \quad t > 0 \quad \text{(II.3)}$$

$$w(t,0) \; = \; w_x(t,0) \; = \; w(t,a) \; = \; w_x(t,a) \; = \; 0\,, \quad t > 0, \tag{II.4}$$

where

$$
\begin{aligned}
w(t,x) \;&= \; \text{the transverse displacement of the beam} \\
\phi(t,x,y) \;&= \; \text{the fluid velocity potential} \\
\rho_b \;&= \; \text{the linear mass density of the beam} \\
\rho_f \;&= \; \text{the uniform mass density of the fluid} \\
M(t,x) \;&= \; EI w_{xx} + c_D I w_{xxt} \\
f(t,x) \;&= \; \text{the force due to an exterior noise field.}
\end{aligned}
$$

For the stability analysis of this model, we consider the open loop problem absent any exterior noise field (i.e., $f(t,x) = 0$ for $t > 0$) and do not concern ourselves with any noise control aspects. We assume (i) that the beam is impenetrable to the adjoining fluid and obtain the second coupling equation (i.e., the continuity of velocity)

$$w_t(x,t) \; = \; \nabla\phi(t,x,w(x,t)) \cdot \hat{n}\,, \quad 0 < x < a\,, \; t > 0\,, \tag{II.5}$$

and (ii) that displacements from the beam's position of rest are small, which is inherent in the Euler-Bernoulli formulation. Because of (ii), we take the transverse displacement of the beam as $w(t,x) = \widetilde{w}(t,x) + \delta$ where $\widetilde{w} \equiv 0$. Under these assumptions, equation II.3 and II.5 become

$$\rho_b w_{tt}(t,x) \; + \; M_{xx}(t,x) \; = \; -\rho_f \phi_t(t,x,\widetilde{w}(t,x) + \delta) \tag{II.6}$$

$$w_t(t,x) \; = \; \nabla\phi(t,x,\widetilde{w}(t,x) + \delta) \cdot \hat{n}\,. \tag{II.7}$$

By using two term Taylor Series expansions of ϕ_t and $\nabla\phi$ with respect to y about the point x, equations II.6 and II.7 become

$$\rho_b w_{tt}(t,x) \; + \; M_{xx}(t,x) \; = \; -\rho_f[\phi_t(t,x,0) \; + \; \phi_{ty}(t,x,0)w]\,,$$

9

$$w_t(t,x) = \nabla\phi(t,x,0)\cdot\hat{n} + (\nabla\phi_y(t,x,0)w)\cdot\hat{n}.$$

We drop the higher order terms $-\rho_f\phi_{ty}(x,0,t)w$ and $(\nabla\phi_y(x,0,t)w)\cdot\hat{n}$ in these two equations because of the assumption of small beam displacement and obtain first order approximations for $-\rho_f\phi_t$ and $\nabla\phi$, respectively. Upon approximating the space domain $\Omega(t)$ by $\Omega \equiv [0,a]\times[0,\ell]$, the open loop model described above is given by

$$\left.\begin{aligned}
\phi_{tt} &= c^2\Delta\phi, \quad (x,y)\in\Omega,\, t>0,\\
\nabla\phi\cdot\hat{n} &= 0, \quad (x,y)\in\Gamma,\, t>0,\\
\phi_y(t,x,0) &= -w_t(t,x), \quad 0<x<a,\, t>0,\\
\rho_b w_{tt} &+ \partial_x^2(EIw_{xx}+c_DIw_{xxt}) = -\rho_f\phi_t(t,x,0),\, 0<x<a,\, t>0,\\
w(t,0) &= w_x(t,0) = w(t,a) = w_x(t,a) = 0, \quad t>0,\\
\phi(0,x,y) &= \phi_0(x,y), \quad w(0,x)=w_0(x),\\
\phi_t(0,x,y) &= \phi_1(x,y), \quad w_t(0,x)=w_1(x).
\end{aligned}\right\} \quad \text{(II.8)}$$

Note: Throughout this paper, ∂_α denotes partial differentiation with respect to the variable α (e.g., $\partial_x^2 = \frac{\partial^2}{\partial x^2}$).

System II.8 is a formal representation of the dynamics of a coupled acoustic/beam structure. Computational techniques (e.g., variational methods) used to obtain approximate solutions to this system are based on rigorous convergence arguments, typically done in the context of variational formulations of II.8. To accomplish this, the state is taken to be $z(t)=(\phi,w)$ in the Hilbert space $H = \overline{L}^2(\Omega)\times L^2(\Gamma_0)$ with energy product

$$\left\langle \begin{pmatrix}\phi\\w\end{pmatrix}, \begin{pmatrix}\xi\\\eta\end{pmatrix} \right\rangle_H = \int_\Omega \frac{\rho_f}{c^2}\phi\xi\,d\omega + \int_{\Gamma_0}\rho_b\,w\eta\,d\gamma,$$

where $\overline{L}^2(\Omega)$ is the quotient space of L^2 over the constant functions. Also, we define the Hilbert space $V = \overline{H}^1(\Omega)\times H_0^2(\Gamma_0)$ where $\overline{H}^1(\Omega)$ is the quotient space of H^1 over the constant functions and $H_0^2(\Gamma_0) = \{\psi \in H^2(\Gamma_0) : \psi(x)=\psi_x(x)=0$ at

$x = 0, a \}$. The energy product of V is taken as

$$\left\langle \begin{pmatrix} \phi \\ w \end{pmatrix}, \begin{pmatrix} \xi \\ \eta \end{pmatrix} \right\rangle_V = \int_\Omega \rho_f \nabla\phi \cdot \nabla\xi \, d\omega + \int_{\Gamma_0} EI w_{xx} \eta_{xx} \, d\gamma \,.$$

Next we define the weak variational (i.e., sesquilinear) forms:

$$\left.\begin{aligned}
\beta_1(\phi, \xi) &= \int_\Omega \rho_f \, \nabla\phi \cdot \nabla\xi \, d\omega \ \text{ for } \ \phi, \xi \in \overline{H}^1(\Omega) \\
\beta_2(w, \eta) &= \int_{\Gamma_0} w_{xx} \eta_{xx} \, d\gamma \ \text{ for } \ w, \eta \in H_0^2(\Gamma_0) \\
\rho_1(w, \eta) &= \int_{\Gamma_0} \rho_b \, w\eta \, d\gamma \ \text{ for } \ w, \eta \in L^2(\Gamma_0) \\
\rho_2(\phi, \xi) &= \int_\Omega \frac{\rho_f}{c^2} \, \phi\xi \, d\omega \ \text{ for } \ \phi, \xi \in \overline{L}^2(\Omega) \\
\mu_1(w, \eta) &= \int_{\Gamma_0} EI \, w_{xx} \eta_{xx} \, d\gamma \ \text{ for } \ w, \eta \in H_0^2(\Gamma_0) \\
\mu_2(\phi, \xi) &= \int_\Omega \rho_f \, \nabla\phi \cdot \nabla\xi \, d\omega \ \text{ for } \ \phi, \xi \in \overline{H}^1(\Omega) \\
\kappa_1(w, \eta) &= \int_{\Gamma_0} c_D I \, w_{xx} \eta_{xx} \, d\gamma \ \text{ for } \ w, \eta \in H_0^2(\Gamma_0) \\
\tau_1(\phi, \eta) &= \int_{\Gamma_0} \rho_f \, \phi(t, x, 0) \eta \, d\gamma \ \text{ for } \ \phi \in \overline{H}^1(\Omega) \text{ and } \eta \in H_0^2(\Gamma_0) \\
\tau_2(w, \xi) &= \int_{\Gamma_0} \rho_f \, \xi(t, x, 0) w \, d\gamma \ \text{ for } \ w \in H_0^2(\Gamma_0) \text{ and } \xi \in \overline{H}^1(\Omega)
\end{aligned}\right\} \qquad \text{(II.9)}$$

and express system II.8 in weak form as,

$$\rho_1(w_{tt}, \eta) + \kappa_1(w_t, \eta) + \mu_1(w, \eta) = -\tau_1(\phi_t, \eta) \ \text{ and } \qquad \text{(II.10)}$$

$$\rho_2(\phi_{tt}, \xi) + \mu_2(\phi, \xi) = \tau_2(w_t, \xi) \,. \qquad \text{(II.11)}$$

Our next task is to write equations II.10 and II.11 as a single second order differential equation. Let $\Phi = (\phi, w)$ and $\Psi = (\xi, \eta)$, such that $\Phi, \Psi \in V$, and define sesquilinear forms:

$$\sigma_1(\Phi, \Psi) = \mu_2 + \mu_1 = \int_\Omega \rho_f \nabla\phi \cdot \nabla\xi \, d\omega + \int_{\Gamma_0} EI w_{xx} \eta_{xx} \, d\gamma \,,$$

$$\sigma_2(\Phi, \Psi) = \kappa_1 + \tau_1 - \tau_2 = \int_{\Gamma_0} \{ c_D I w_{xx} \eta_{xx} + \rho_f(\phi(t, x, 0)\eta - \xi(t, x, 0)w \} \, d\gamma \,,$$

where $\sigma_1, \sigma_2 \in V \times V \longrightarrow \mathcal{C}$ (space of complex numbers). The formulations σ_1 and σ_2 satisfy coercivity and continuity (i.e., boundedness) conditions

$$\Re \, \sigma_1(\Phi, \Phi) \geq c_1 \|\Phi\|_V^2 \,,$$

$$\|\sigma_1(\Phi, \Psi)\| \leq c_2 \|\Phi\|_V \|\Psi\|_V \,,$$

$$\Re \, \sigma_2(\Phi, \Phi) \geq c_3 \langle w_{xx}, w_{xx} \rangle_{L^2(\Gamma_0)} = c_3 \|w\|^2_{H^2_0(\Gamma_0)} \,,$$

$$\|\sigma_2(\Phi, \Psi)\| \leq c_4 \|\Phi\|_V \|\Psi\|_V \,,$$

where $\Re$ denotes the "real part of". The second order open loop problem is given by

$$\langle z_{tt}(t), \Psi \rangle_{V^*, V} + \sigma_2(z_t(t), \Psi) + \sigma_1(z(t), \Psi) = 0 \,, \tag{II.12}$$

where V^* is the dual of space V and $\langle \cdot, \cdot \rangle_{V^*, V}$ denotes the usual duality pairing.

Since σ_1 and σ_2 satisfy the continuity and coercivity conditions shown above, the Lax-Milgram Theorem guarantees the existence of uniquely determined bounded linear operators A_1, A_2 such that the weak and strong formulations of the coupled system are equivalent [Ref. 8]. That is

$$\langle A_1 \Phi, \Psi \rangle_{V^*, V} = \sigma_1(\Phi, \Psi) \quad \text{and} \quad \langle A_2 \Phi, \Psi \rangle_{V^*, V} = \sigma_2(\Phi, \Psi) \,.$$

Thus system II.12 gives rise to the equivalent system defined in terms of functionals A_1 and A_2

$$z_{tt}(t) + A_2 z_t(t) + A_1 z(t) = 0 \,. \tag{II.13}$$

To facilitate our numerical work, we must express system II.8 as a first order system. Our goal is to write Model I as

$$\underbrace{\begin{bmatrix} \phi_t \\ w_t \\ \phi_{tt} \\ w_{tt} \end{bmatrix}}_{u_t} = \underbrace{\begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ c^2 \Delta & 0 & 0 & 0 \\ 0 & -\frac{EI}{\rho_b} \partial_x^4 & -\Pi & -\frac{c_D I}{\rho_b} \partial_x^4 \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} \phi \\ w \\ \phi_t \\ w_t \end{bmatrix}}_{u} . \tag{II.14}$$

The symbol Π appearing in matrix $\mathcal{A}$ above represents $\frac{\rho_f}{\rho_b} \phi_t(t, x, 0)$ whenever the product $\mathcal{A}u$ is calculated.

We now develop the necessary notation and sesquilinear forms in order to write system II.8 as a first order system. The following approach replicates that found in [Ref. 9].

12

We begin by defining the product spaces $\mathcal{V} = V \times V$ and $\mathcal{H} = V \times H$ whose norms are given by

$$\|(\Phi, \Psi)\|_{\mathcal{V}}^2 = \|\Phi\|_V^2 + \|\Psi\|_V^2 \quad \text{and} \quad \|(\Phi, \Psi)\|_{\mathcal{H}}^2 = \|\Phi\|_V^2 + \|\Psi\|_H^2 \, , \quad \text{respectively.}$$

For $\chi = (\Phi, \Psi)$ and $\Theta = (\Upsilon, \Lambda)$, the sesquilinear form $\sigma : \mathcal{V} \times \mathcal{V} \longrightarrow \mathcal{C}$ is then defined by

$$\sigma(\Theta, \chi) = \sigma((\Upsilon, \Lambda), (\Phi, \Psi)) = -\langle \Lambda, \Phi \rangle_V + \sigma_1(\Upsilon, \Psi) + \sigma_2(\Lambda, \Psi) \, .$$

Since the duality product $\langle \cdot, \cdot \rangle_{V^*, V}$ is the unique extension by continuity of the scalar product $\langle \cdot, \cdot \rangle_H$ from $H \times V$ to $V^* \times V$, it follows that for appropriate restrictions on Θ we can write

$$
\begin{aligned}
\sigma(\Theta, \chi) = \sigma((\Upsilon, \Lambda), (\Phi, \Psi)) &= -\langle \Lambda, \Phi \rangle_V + \langle A_1 \Upsilon, \Psi \rangle_{V^*, V} + \langle A_2 \Lambda, \Psi \rangle_{V^*, V} \\
&= -\langle \Lambda, \Phi \rangle_V + \langle A_1 \Upsilon + A_2 \Lambda, \Psi \rangle_H \\
&= \langle (-\Lambda, A_1 \Upsilon + A_2 \Lambda), (\Phi, \Psi) \rangle_{\mathcal{H}} \\
&= \langle -\mathcal{A}\Theta, \chi \rangle_{\mathcal{H}} \, .
\end{aligned}
$$

The operator $\mathcal{A} : \mathcal{H} \longrightarrow \mathcal{H}$ is given by

$$\mathcal{A} = \begin{bmatrix} 0 & I \\ -A_1 & -A_2 \end{bmatrix} \, , \tag{II.15}$$

where the domain of $\mathcal{A} = \{\Theta = (\Upsilon, \Lambda) \in \mathcal{H} : \Lambda \in V, A_1 \Upsilon + A_2 \Lambda \in H\}$, A_1 and A_2 are the operators defined by σ_1 and σ_2, respectively, and the above calculations hold for $\Theta \in$ domain $\mathcal{A}$.

By letting $Z(t) = (z(t), z_t(t))$ and taking $\chi \in \mathcal{V}$, the first order system is written in weak form as

$$\langle Z_t(t), \chi \rangle_{\mathcal{V}^*, \mathcal{V}} = -\sigma(Z(t), \chi) \, ,$$

which is formally equivalent to the strong formulation

$$Z_t(t) = \mathcal{A}Z(t) \tag{II.16}$$

13

in $\mathcal{H}$, where $\mathcal{A}$ is given in II.15. Equation II.16 concisely represents the matrix representation of Model I given by equation II.14 where

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ c^2\Delta & 0 & 0 & 0 \\ 0 & -\frac{EI}{\rho_b}\partial_x^4 & -\Pi & -\frac{c_D I}{\rho_b}\partial_x^4 \end{bmatrix}, \text{ and}$$

$$A_1 = \begin{bmatrix} -c^2\Delta & 0 \\ 0 & \frac{EI}{\rho_b}\partial_x^4 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 \\ \Pi & \frac{c_D I}{\rho_b}\partial_x^4 \end{bmatrix}.$$

The linear operator $\mathcal{A}$ is dissipative in the sense $\langle \mathcal{A}\chi, \chi \rangle_{\mathcal{H}} \leq 0$ for $\chi \in$ domain $\mathcal{A}$. To

see this let $\chi = (\phi, w, \phi_t, w_t)$, then

$$\langle \mathcal{A}\chi, \chi \rangle_{\mathcal{H}} = \left\langle \begin{bmatrix} \phi_t \\ w_t \\ c^2 \Delta \phi \\ -\frac{EI}{\rho_b} \partial_x^4 w - \Pi \phi_t - \frac{c_D I}{\rho_b} \partial_x^4 w_t \end{bmatrix}, \begin{bmatrix} \phi \\ w \\ \phi_t \\ w_t \end{bmatrix} \right\rangle_{\mathcal{H}=V \times H}$$

$$= \langle \rho_f \nabla \phi_t, \nabla \phi \rangle_{L^2(\Omega)} + \langle EI w_{xxt}, w_{xx} \rangle_{L^2(\Gamma_0)} + \langle \rho_f \Delta \phi, \phi_t \rangle_{L^2(\Omega)}$$

$$- \underbrace{\langle EI w_{xx}, w_{xxt} \rangle_{L^2(\Gamma_0)}}_{\text{by integration by parts}} - \langle \rho_f \phi_t(t,x,0), w_t \rangle_{L^2(\Gamma_0)} - \underbrace{\langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)}}_{\text{by integration by parts}}$$

$$= \langle \rho_f \nabla \phi_t, \nabla \phi \rangle_{L^2(\Omega)} + \underbrace{\langle \rho_f \Delta \phi, \phi_t \rangle_{L^2(\Omega)}}_{\text{apply Green's formula}} - \langle \rho_f \phi_t(t,x,0), w_t \rangle_{L^2(\Gamma_0)}$$

$$- \langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)}$$

$$= \langle \rho_f \nabla \phi_t, \nabla \phi \rangle_{L^2(\Omega)} + \langle \rho_f \partial_n \phi, \phi_t \rangle_{L^2(\Gamma_0)} - \langle \rho_f \nabla \phi, \nabla \phi_t \rangle_{L^2(\Omega)}$$

$$- \langle \rho_f \phi_t(t,x,0), w_t \rangle_{L^2(\Gamma_0)} - \langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)}$$

$$= \underbrace{\langle \rho_f \partial_n \phi, \phi_t \rangle_{L^2(\Gamma_0)}}_{\text{apply } \partial_n \phi = -w_t \text{ on } \Gamma_0} - \langle \rho_f \phi_t(t,x,0), w_t \rangle_{L^2(\Gamma_0)} - \langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)}$$

$$= \langle \rho_f w_t, \phi_t \rangle_{L^2(\Gamma_0)} - \langle \rho_f \phi_t(t,x,0), w_t \rangle_{L^2(\Gamma_0)} - \langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)}$$

$$= -\langle c_D I w_{xxt}, w_{xxt} \rangle_{L^2(\Gamma_0)} \, .$$

$$\left. \right\} \qquad (\text{II.17})$$

Because $-c_D I \|w_{xxt}\|_{L^2(\Gamma_0)}^2 \leq 0$, we see that $\mathcal{A}$ is in fact dissipative.

B. MATHEMATICAL MODEL II
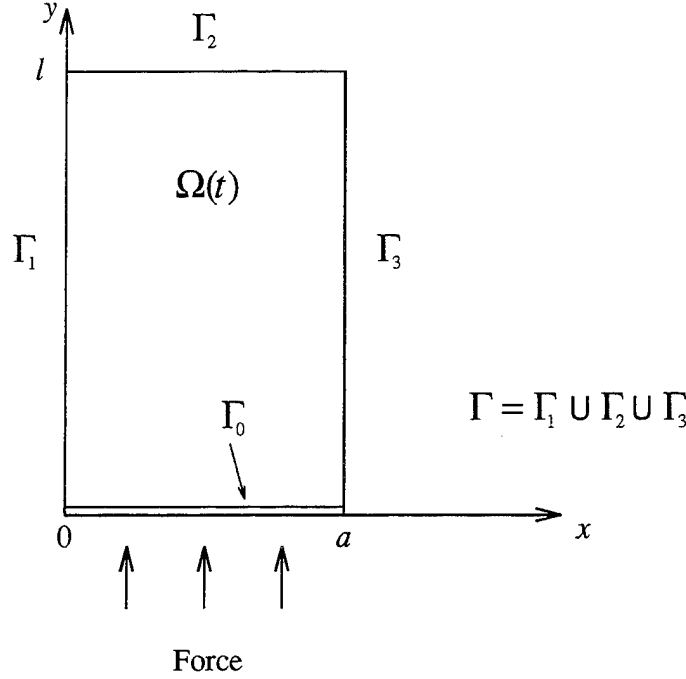
Figure 3. 2-D Acoustic Chamber for Model II

Our second model is similar to the first in that we consider the open loop case in which the wave equation is coupled with the beam equation on one side of a 2-D rectangular, air filled cavity as shown in Figure 3. Let $\Omega = [0, a] \times [0, l]$, $\partial\Omega = \Gamma_0 \cup \Gamma$ with $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. Let the velocity potential in Ω be given by $\phi = \phi(t, x, y)$ and the transverse beam displacement be denoted $w = w(t, x)$ as in Model I. Once again we take our perturbable boundary to be a fixed-end (i.e., clamped) Euler Bernoulli beam with Kelvin-Voigt damping. However, instead of assuming hard wall boundary conditions along Γ as we did in the first model, we now consider dissipative boundary conditions along Γ. Specifically, the boundary conditions are taken to be

$$\partial_n \phi + a\phi_t = 0 \text{ for } (x, y) \in \Gamma \text{ and } \partial_n \phi = w_t(t, x) \text{ for } (x, 0) \in \Gamma_0, \ t > 0, \qquad \text{(II.18)}$$

where a is a constant of proportionality. As in Model I, we take $f(t, x) = 0$ for $t > 0$. All other assumptions remain as stated in Model I. Hence the coupled system for

16

Model II is given by

$$
\begin{aligned}
\phi_{tt} &= c^2 \Delta\phi, \quad (x,y) \in \Omega,\, t > 0, \\
\partial_n\phi &= -a\phi_t, \quad (x,y) \in \Gamma,\, t > 0, \\
\partial_n\phi &= w_t(t,x), \quad (x,0) \in \Gamma_0,\, t > 0, \\
\rho_b w_{tt} &+ \partial_x^2(EIw_{xx} + c_D I w_{xxt}) = -\rho_f\phi_t(t,x,0),\, 0 < x < a,\, t > 0, \\
w(t,0) &= w_x(t,0) = w(t,a) = w_x(t,a) = 0, \quad t > 0, \\
\phi(0,x,y) &= \phi_0(x,y) \quad, \quad w(0,x) = w_0(x), \\
\phi_t(0,x,y) &= \phi_1(x,y) \quad, \quad w_t(0,x) = w_1(x).
\end{aligned}
\right\} \quad \text{(II.19)}
$$

The Hilbert spaces $H = \overline{L}^2(\Omega) \times L^2(\Gamma_0)$ and $V = \overline{H}^1(\Omega) \times H_0^2(\Gamma_0)$ remain as defined in the previous section.

In terms of the sesquilinear forms given by II.9 and

$$
\lambda_1(\phi, \xi) = \int_\Gamma a\,\rho_f\,\phi\xi\,d\gamma \quad \text{for} \quad \phi, \xi \in \overline{L}^2(\Omega),
$$

the weak variational form of II.19 is given by

$$
\rho_1(w_{tt}, \eta) + \kappa_1(w_t, \eta) + \mu_1(w, \eta) = -\tau_1(\phi_t, \eta) \quad \text{and} \tag{II.20}
$$

$$
\rho_2(\phi_{tt}, \xi) + \lambda_1(\phi_t, \xi) + \mu_2(\phi, \xi) = \tau_2(w_t, \xi). \tag{II.21}
$$

Here we pause to explain how the boundary condition $\partial_n\phi + a\phi_t = 0$ gives rise to the sesquilinear form $\lambda_1(\phi, \xi)$. Consider the wave equation $\phi_{tt} = c^2\Delta\phi$. By multiplying through by an appropriately chosen trial function ξ and integrating over Ω we obtain

$$
\int_\Omega \frac{\rho_f}{c^2}\,\phi_{tt}\xi\,d\omega = \int_\Omega \rho_f\,\Delta\phi\,\xi\,d\omega.
$$

Looking just at the right hand side of this equation, we use Green's formula to obtain

$$
\int_\Omega \rho_f\,\Delta\phi\xi\,d\omega = \int_{\partial\Omega} \rho_f\,(\partial_n\phi)\xi\,d\omega - \int_\Omega \rho_f\,\nabla\phi \cdot \nabla\xi\,d\omega \quad \text{where} \quad \partial\Omega = \Gamma \cup \Gamma_0.
$$

The boundary term represents

$$
\int_{\partial\Omega} \rho_f\,(\partial_n\phi)\xi\,d\omega = \int_{\Gamma_0} \rho_f\,w_t\xi\,d\gamma + \int_\Gamma \rho_f\,(-a)\,\phi_t\xi\,d\gamma = \tau_2(w_t, \xi) - \lambda_1(\phi_t, \xi).
$$

The only difference between the weak formulations for Models I and II (equations II.10, II.11 and II.20, II.21 respectively) is the appearance of $\lambda_1(\phi, \xi)$ in equation II.21 which, as we have seen, arises because of the damping along Γ. Just as the sesquilinear forms were shown to satisfy certain coercivity and continuity conditions in the previous section, so too does $\lambda_1(\phi, \xi)$. Once again, the Lax-Milgram Theorem assures us that these weak formulations give rise to uniquely determined bounded linear operators. Hence Model II can be expressed as a single first order equation analogous to equation II.16 in the previous section.

Letting $\Phi = (\phi, w)$ and $\Psi = (\xi, \eta)$, such that $\Phi, \Psi \in V$, and define sesquilinear forms:

$$\sigma_1(\Phi, \Psi) = \int_\Omega \rho_f \nabla \phi \cdot \nabla \xi \, d\omega + \int_{\Gamma_0} EI w_{xx} \eta_{xx} \, d\gamma \,,$$

$$\sigma_2(\Phi, \Psi) = \int_{\Gamma_0} \{ c_D I w_{xx} \eta_{xx} + \rho_f(\phi(t,x,0)\eta - \xi(t,x,0)w) \} \, d\gamma + \int_\Gamma a \rho_f \, \phi \xi \, d\gamma,$$

where $\sigma_1, \sigma_2 \in V \times V \longrightarrow \mathcal{C}$ (space of complex numbers).

Once again, the formulations σ_1 and σ_2 satisfy coercivity and continuity conditions

$$\Re \sigma_1(\Phi, \Phi) \geq c_1 \|\Phi\|_V^2 \,,$$

$$\|\sigma_1(\Phi, \Psi)\| \leq c_2 \|\Phi\|_V \|\Psi\|_V \,,$$

$$\Re \sigma_2(\Phi, \Phi) \geq c_3 \langle w_{xx}, w_{xx} \rangle_{L^2(\Gamma_0)} + c_3 \|\phi\|_{L^2(\Gamma)}^2 = c_3 \|w\|_{H_0^2(\Gamma_0)}^2 + c_3 \|\phi\|_{L^2(\Gamma)}^2 \,,$$

$$\|\sigma_2(\Phi, \Psi)\| \leq c_4 \|\Phi\|_V \|\Psi\|_V \,.$$

Denoting our state variables by $z(t) = (\phi, w)$ and making use of σ_1 and σ_2 as defined above, we can express the second order open loop problem concisely as

$$\langle z_{tt}(t), \Psi \rangle_{V^*, V} + \sigma_2(z_t(t), \Psi) + \sigma_1(z(t), \Psi) = 0 \,, \tag{II.22}$$

where V^* is the dual of space V.

Associated with σ_1 and σ_2 are functionals A_1, A_2 such that the weak and strong formulations of the coupled system are equivalent. That is

$$\langle A_1 \Phi, \Psi \rangle_{V^*, V} = \sigma_1(\Phi, \Psi) \text{ and } \langle A_2 \Phi, \Psi \rangle_{V^*, V} = \sigma_2(\Phi, \Psi) \,.$$

18

We conclude that the system described by equation II.22 gives rise to an equivalent system defined in terms of operators A_1 and A_2. That is

$$z_{tt}(t) + A_2 z_t(t) + A_1 z(t) = 0. \tag{II.23}$$

The first order system is obtained just as it is for Model I. We let $Z(t) = (z(t), z_t(t))$ and take $\chi \in \mathcal{V}$ to obtain

$$\langle Z_t(t), \chi \rangle_{\mathcal{V}^*, \mathcal{V}} = -\sigma(Z(t), \chi).$$

Formally, this system is equivalent to the strong formulation

$$Z_t(t) = \mathcal{A}Z(t) \tag{II.24}$$

in $\mathcal{H}$. In matrix form, the strong formulation Model II is

$$\underbrace{\begin{bmatrix} \phi_t \\ w_t \\ \phi_{tt} \\ w_{tt} \end{bmatrix}}_{u_t} = \underbrace{\begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ c^2\Delta & 0 & 0 & 0 \\ 0 & -\frac{EI}{\rho_b}\partial_x^4 & -\Pi & -\frac{c_D I}{\rho_b}\partial_x^4 \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} \phi \\ w \\ \phi_t \\ w_t \end{bmatrix}}_{u}, \tag{II.25}$$

where the block matrices A_1, A_2 of $\mathcal{A}$ are

$$A_1 = \begin{bmatrix} -c^2\Delta & 0 \\ 0 & \frac{EI}{\rho_b}\partial_x^4 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 0 & 0 \\ \Pi & \frac{c_D I}{\rho_b}\partial_x^4 \end{bmatrix}.$$

Since energy is lost along all four sides of the cavity in Model II, it is actually more dissipative than Model I, where energy is lost only along one side. Following the procedure shown in II.17 one finds that

$$\langle \mathcal{A}\chi, \chi \rangle_{\mathcal{H}} = -c_D I \|w_{xxt}\|_{L^2(\Gamma_0)} - \mathsf{a}\langle \rho_f \, \phi_t, \phi_t \rangle_{L^2(\Gamma)} = -c_D I \|w_t\|_{H_0^2(\Gamma_0)}^2 - \mathsf{a}\rho_f \|\phi_t\|_{L^2(\Gamma)}^2 \le 0,$$

where a is the proportionality constant which appears in boundary condition II.18.

Now that we have variational formulations for Models I and II, we turn our attention to a numerical scheme often used to obtain approximate solutions to systems of differential equations.

19

III.   GALERKIN APPROXIMATION METHOD

In order to examine the exponential stability of approximate solutions to Models I and II, we employ the Galerkin method to transform the systems of partial differential equations into systems of ordinary differential equations. A review of this popular computational technique follows.

The Galerkin method is one of the variational methods (e.g., Rayleigh-Ritz, Least Squares, Galerkin, Collocation, etc.) which all seek good approximate solutions to many types of boundary-value and initial-boundary value problems of the form $Au = f$ from a finite dimensional subspace.

Specifically, when given the differential equation $Au = f$, where A maps the normed linear space X with norm $\| \cdot \|_X$ into the normed linear space Y with norm $\| \cdot \|_Y$ and a finite dimensional subspace $X_N = span\{\phi_1, \phi_2, ..., \phi_N\}$ of X, variational methods seek functions

$$u_N = c_1\phi_1 + c_2\phi_2 + ... + c_N\phi_N$$

belonging to X_N which minimize

$$\|Au_N - f\|_Y + \|u_N - u\|_X.$$

The Galerkin method is a widely used technique, subsuming both the finite element method and the method of least squares. It is worth noting that the Galerkin and Rayleigh-Ritz methods coincide whenever the differential operator A is linear, positive definite and self-adjoint. However, there are many important differential operators which are either nonlinear, not self-adjoint or non-symmetric for which the Galerkin method is applicable while the Rayleigh-Ritz method fails. For example, the Galerkin scheme is applicable to many parabolic and hyperbolic differential equations whereas the Rayleigh-Ritz method may not be since the associated variational problem may not have a solution [Ref. 10].

The Galerkin method yields a finite system of equations, from a differential equation, by discretizing the solution space rather than by discretizing the domain and operators, as is done in other popular methods. Approximate solutions are found using a variant of the method of weighted residuals (MWR), where the weighting functions are chosen to yield solutions which are a finite combination of known functions. The task at hand is to determine approximate solutions to the linear (or nonlinear) differential operator equation

$$Au = f$$

from a finite dimensional subspace X_N of some inner product space X in which the operator A is defined. Let $(\cdot, \cdot)$ denote an inner product on X, let $X_N = \text{span}\{\phi_1^N, \phi_2^N, ..., \phi_N^N\}$, and let $Y_N = \text{span}\{\psi_1^N, \psi_2^N, ..., \psi_N^N\}$ where X_N, Y_N are N-dimensional subspaces of X. The MWR seeks an approximate solution u_N to $Au = f$ such that $u_N \in X_N$ satisfies the system of equations

$$(Au_N - f, \psi_j^N) = 0, \text{ for } j = 1, 2, ..., N. \tag{III.1}$$

Equation III.1 is the inner product of the residual $(Au_N - f)$ and the weighting function ψ_j^N integrated over the appropriate region. Now take u_N to be a linear combination of the basis functions ϕ_i^N such that

$$u_N = c_1 \phi_1^N + c_2 \phi_2^N + ... + c_N \phi_N^N.$$

Upon substituting this expression for u_N into equation III.1, we find that u_N must satisfy the linear system

$$\sum_{i=1}^{N} (A\phi_i^N, \psi_j^N) c_i = (f, \psi_j^N), \text{ for } j = 1, 2, ..., N.$$

In the Galerkin scheme, the weighting functions ψ_j^N are taken to be the basis functions of the approximate solution. That is

$$\psi_j^N = \phi_j^N \text{ for } j = 1, 2, ..., N.$$

The approximate solution is obtained by solving the resulting system for the coefficient functions $c_i(t)$.

An example will clarify the general procedure. Consider the following initial-boundary value problem

$$u_t - k u_{xx} = 0, \quad \text{over } \Gamma = [0,1], \quad t \geq 0, \tag{III.2}$$

$$u(t,0) = 0, \quad u_x(t,1) = 0, \tag{III.3}$$

$$u(0,x) = x, \tag{III.4}$$

describing the heat conduction through a thin one-dimensional rod with no sources of thermal energy. The temperature is held constant at one end of the rod (i.e., at $x = 0$); the other end is insulated. Here we take the thermal diffusivity of the rod, k, to equal one.

In operator notation, equation III.2 can be written as

$$u_t = Lu$$

where the operator $L = k \partial_x^2$ and, for simplicity, we take $k = 1$.

Our first task is to select appropriate basis functions which (i) possess the smoothness requirements of the second order problem (i.e., $\phi_i^N \in C^2(\Gamma)$) , (ii) are linearly independent on Γ, and (iii) satisfy the boundary conditions III.3. Here we choose

$$\phi_i^N = x \left(\frac{x^i}{i+1} - 1 \right).$$

For this example we take $N = 2$ obtaining

$$\phi_1^N = \frac{x^2}{2} - x \quad \text{and} \quad \phi_2^N = \frac{x^3}{3} - x,$$

and because we will need the first and second derivatives of these basis functions later, we calculate them now:

$$\partial_x \phi_1^N = x - 1, \quad \partial_x^2 \phi_1^N = 1,$$
$$\partial_x \phi_2^N = x^2 - 1, \quad \partial_x^2 \phi_2^N = 2x.$$

Our approximate solution takes the form

$$u(t,x) \sim u_N(t,x) = \sum_{i=1}^{N=2} c_i(t)\phi_i^N(x) = c_1(t)\phi_1^N(x) + c_2(t)\phi_2^N(x),$$

where we seek functions $c_i(t)$. For $N = 2$, we need to select two weighting functions ω_1, ω_2 and introduce the spacial average (i.e., inner product or weighted integral)

$$(\omega, v) = \int_\Gamma \omega v \, d\gamma,$$

such that

$$(\omega_i, u_t) = (\omega_i, Lu_N) \quad \text{for } i = 1, 2. \tag{III.5}$$

Note: The spacial superscript N is omitted below where the meaning is clear and an overdot denotes differentiation in time.

For the Galerkin scheme, the basis functions ϕ_i serve as the weighting functions. Hence III.5 becomes

$$(\phi_i, \dot{u}_N) = (\phi_i, Lu_N) \quad \text{for } i = 1, 2$$

yielding the system

$$(\phi_1, \dot{c}_1\phi_1) + (\phi_1, \dot{c}_2\phi_2) - (\phi_1, c_1\partial_x^2\phi_1) - (\phi_1, c_2\partial_x^2\phi_2) = 0,$$

$$(\phi_2, \dot{c}_1\phi_1) + (\phi_2, \dot{c}_2\phi_2) - (\phi_2, c_1\partial_x^2\phi_1) - (\phi_2, c_2\partial_x^2\phi_2) = 0,$$

or equivalently,

$$\begin{bmatrix} (\phi_1, \phi_1) & (\phi_1, \phi_2) \\ (\phi_2, \phi_1) & (\phi_2, \phi_2) \end{bmatrix} \begin{bmatrix} \dot{c}_1 \\ \dot{c}_2 \end{bmatrix} - \begin{bmatrix} (\phi_1, \partial_x^2\phi_1) & (\phi_1, \partial_x^2\phi_2) \\ (\phi_2, \partial_x^2\phi_1) & (\phi_2, \partial_x^2\phi_2) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{III.6}$$

Given the particular boundary conditions for this problem, and because the basis functions ϕ_i we selected satisfy these boundary conditions, we find that the inner

product $(\phi_i, \partial_x^2 \phi_j)$ is equivalent to the inner product $-(\partial_x \phi_i, \partial_x \phi_j)$ (i.e., L is a self-adjoint operator). To see this, we integrate by parts:

$$(\phi_i, \partial_x^2 \phi_j) = \int_0^1 \phi_i \partial_x^2 \phi_j \, dx = \phi_i \partial_x \phi_j \big|_0^1 - \int_0^1 \partial_x \phi_i \cdot \partial_x \phi_j \, dx \,,$$

where the boundary term vanishes. Thus,

$$(\phi_i, \partial_x^2 \phi_j) = -(\partial_x \phi_i, \partial_x \phi_j) \,.$$

Although the basis functions ϕ_i were initially chosen so that they belonged to $C^2(\Gamma)$, this result indicates that less regular functions may suffice for this problem (i.e, $\phi_i \in C^1(\Gamma)$). This fact, which is of little use in this simple example, is emphasized because of its theoretical import as well as its utility in reducing the computational complexity of more challenging problems.

We now calculate the inner products contained in equation III.6 to obtain

$$\underbrace{\begin{bmatrix} \frac{2}{15} & \frac{61}{360} \\ \frac{61}{360} & \frac{68}{315} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \dot{c}_1 \\ \dot{c}_2 \end{bmatrix}}_{\vec{c}} + \underbrace{\begin{bmatrix} \frac{1}{3} & \frac{5}{12} \\ \frac{5}{12} & \frac{8}{15} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{\vec{c}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\vec{0}} . \tag{III.7}$$

Thus the Galerkin method has reduced our problem of finding approximate solutions to a second order *partial* differential equation (PDE) to that of finding approximate solutions to a system of first order *ordinary* differential equations (ODEs). A much simpler task indeed! Multiplying equation III.7 by M^{-1} yields

$$\begin{bmatrix} \dot{c}_1 \\ \dot{c}_2 \end{bmatrix} + \begin{bmatrix} 18.9231 & -5.9077 \\ -12.9231 & 7.1077 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} ,$$

or equivalently,

$$\dot{c}_1 + 18.9231 c_1 - 5.9077 c_2 = 0 \,, \tag{III.8}$$

$$\dot{c}_2 - 12.9231 c_1 + 7.1077 c_2 = 0 \,. \tag{III.9}$$

To determine c_1 and c_2, we first solve for c_2 in equation III.9 and then differentiate to obtain $\dot{c}_2$ yielding

$$c_2 = \frac{1}{5.9077}(\dot{c}_1 + 18.9231c_1), \tag{III.10}$$

$$\dot{c}_2 = \frac{1}{5.9077}(\ddot{c}_1 + 18.9231\dot{c}_1). \tag{III.11}$$

Next substitute these equations for c_2 and $\dot{c}_2$ into equation III.9 and simplify to obtain

$$\ddot{c}_1 + 26.0308\dot{c}_1 + 58.1539c_1 = 0.$$

This is a second order ODE with constant coefficients which is easily solved. We find that the general solutions for $c_1, \dot{c}_1$ are

$$c_1 = \alpha e^{-2.4681t} + \beta e^{-23.5628t},$$

$$\dot{c}_1 = -2.4681\alpha e^{-2.4681t} - 23.5628\beta e^{-23.5628t},$$

where α, β are constants. Substituting these equations for $c_1, \dot{c}_1$ into equation III.10 we obtain

$$c_2 = 2.7853\alpha e^{-2.4681t} - 0.7854\beta e^{-23.5628t}. \tag{III.12}$$

In order to determine α and β, we make use of the given initial condition given by equation III.4. For $(t = 0, x = .5)$ and $(t = 0, x = 1)$, we know that $u(0, .5) = .5$ and $u(0, 1) = 1$, respectively. Hence we have

$$u(0, .5) \sim u_N(0, .5) = c_1(0)\phi_1(.5) + c_2(0)\phi_2(.5) = .5$$
$$-1.6515\alpha - 0.0151\beta = .5$$

and

$$u(0, 1) \sim u_N(0, 1) = c_1(0)\phi_1(1) + c_2(0)\phi_2(1) = 1$$
$$-2.3569\alpha + 0.0236\beta = 1.$$

From these two equations, we determine that

$$\alpha = -0.3608 \quad \text{and} \quad \beta = 6.3442.$$

Therefore, our approximate solution to equation III.2 is

$$u_N(t, x) = \sum_{i=1}^{N=2} c_i(t)\phi_i(x),$$

where

$$c_1 = -0.3608e^{-2.4681t} + 6.3442e^{-23.5628t}$$

$$c_2 = -1.0049e^{-2.4681t} - 4.9827e^{-23.5628t}$$

$$\phi_1 = \frac{x^2}{2} - x$$

$$\phi_2 = \frac{x^3}{3} - x .$$

In summary, the Galerkin method simplified the task of finding an approximate solution of a PDE by recasting the problem as a system of ODEs in a finite dimensional space. Because this example was simply meant to illustrate the Galerkin technique, the solution obtained provides only a very crude approximate solution to the example problem. Techniques for refinement of the approximate solution typically include such things as increasing the number of basis functions and/or selection of different basis functions. The interested reader is referred to [Ref. 11, 12, 13] for additional information regarding solution refinement techniques. In Chapter IV, we transform Models I and II into systems of finite dimension using the Galerkin method.

IV. FINITE DIMENSIONAL APPROXIMATIONS

In Chapter II, both strong and weak variational forms were obtained for Models I and II. Formally, at least, we demonstrated that these two formulations were equivalent. In Chapter III, we saw how approximate solutions to an infinite dimensional PDE could be obtained by employing the Galerkin technique to formulate the problem in finite dimensional spaces. Key steps in implementing this particular discretization technique included selection of an appropriately defined set of basis functions, use of these basis functions as the weighting (or trial) functions, calculation of the inner product(s) defined for the discretization space, and finally, integration over the spacial domain to transform the system of PDEs into a system of time dependent ODEs. In this Chapter we extend these ideas to coupled systems of PDEs, to wit, Models I and II.

Our approach will be: (i) choose finite sets of basis functions which span the approximating solution spaces, (ii) express the infinite dimensional state variables $(w(t, x), \phi(t, x, y))$ in terms of these basis functions, and (iii) use of the weak variational forms developed in Chapter II to obtain finite dimensional representations of Models I and II necessary for our numerical work.

First, let $\{B_i^n\}_{i=1}^{n-1}$ denote the 1-D basis functions which discretize the beam and let $\{B_k^m\}_{k=1}^{m}$ denote the 2-D basis functions which discretize the cavity. For the moment simply note that there are $n-1$ basis functions in $\{B_i^n\}_{i=1}^{n-1}$ and $m = (m_x+1)(m_y+1)-1$ basis functions in $\{B_k^m\}_{k=1}^{m}$ where (m_x+1) and (m_y+1) represent the number of basis functions discretizing the cavity along the x, y axes, respectively. In Chapter V, when we look at specific finite dimensional approximations of Models I and II, the reader will better appreciate why these spaces have the dimensions given.

The basis sets $\{B_i^n\}_{i=1}^{n-1}$ and $\{B_k^m\}_{k=1}^{m}$ span spaces H_b^n and H_c^m where the subscripts b and c denote 'beam' and 'cavity'. That is $H_b^n = \text{span}\{B_i^n\}_{i=1}^{n-1}$ and

$H_c^m = \text{span}\{B_k^m\}_{k=1}^m$. Denoting the combined dimension of the discretized beam and cavity spaces as $N = m + (n-1)$, the approximating beam and cavity solutions are given by

$$w^N(t,x) = \sum_{i=1}^{n-1} w_i^N(t)B_i^n(x) \text{ and } \phi^N(t,x,y) = \sum_{k=1}^m \phi_k^N(t)B_k^m(x,y).$$

Notice that in this form the state variables are separated into products of time and spatial functions.

For application of the Galerkin scheme, elements of the basis sets $\{B_k^m\}_{k=1}^m$ and $\{B_i^n\}_{i=1}^{n-1}$ serve as weighting functions for the cavity and beam, respectively. Denoting the product space for the first order system as $\mathcal{H}^N = H^N \times H^N$, restriction of the infinite dimensional first order systems obtained for Models I and II in Chapter II to the space $\mathcal{H}^N \times \mathcal{H}^N$ yields

$$\langle Z_t^N(t), \chi \rangle_{\mathcal{H}} = -\sigma(Z^N(t), \chi),$$

for $Z^N(t) = (\phi^N(t,x,y), w^N(t,x), \phi_t^N(t,x,y), w_t^N(t,x,y))^T$. Note that σ is model specific (Recall that we used σ to denote the first order sesquilinear vectors for both Models I and II in Chapter II.). For $\chi = (B_\ell^m, B_j^n)$, this finite dimensional first order equation represents the linear system

$$M^N \dot{y}^N(t) = \mathcal{A}^N y^N(t), \tag{IV.1}$$

where

$$y^N = (\vartheta^N(t), \dot{\vartheta}^N(t))^T \text{ and } \vartheta^N = (\phi_1^N(t), \phi_2^N(t), \cdots, \phi_m^N(t), w_1^N(t), w_2^N(t), \cdots, w_{n-1}^N(t))^T$$

denotes the $N \times 1 = (m + n - 1)$ approximate state vector. We use an overdot to denote differentiation with respect to time.

Note: Below, and for the remainder of this paper, the index ranges are $k, \ell = 1, \cdots, m$ and $i, j = 1, \cdots, n - 1$ unless otherwise noted.

Up to this point, everything we have discussed in this chapter applies both to Models I and II. Now we restrict our discussion to Model I for which equation

IV.1 represents the linear system shown below. The non-zero entries in M^N and $\mathcal{A}^N$ represent block matrices. The rows of these block matrices are generated by holding ℓ and j fixed while varying k and i as appropriate for the particular product being computed. System IV.1 is given by

$$
\underbrace{\begin{bmatrix}
\beta_1\left(\sum_{k=1}^m B_k^m, B_\ell^m\right) & 0 & 0 & 0 \\
0 & \beta_2\left(\sum_{i=1}^{n-1} B_i^n, B_j^n\right) & 0 & 0 \\
0 & 0 & \rho_2\left(\sum_{k=1}^m B_k^m, B_\ell^m\right) & 0 \\
0 & 0 & 0 & \rho_1\left(\sum_{i=1}^{n-1} B_i^n, B_j^n\right)
\end{bmatrix}}_{M^N}
\underbrace{\begin{bmatrix}
\dot{\phi}_k(t) \\
\dot{w}_i(t) \\
\ddot{\phi}_k(t) \\
\ddot{w}_i(t)
\end{bmatrix}}_{\dot{y}^N(t)} =
$$

$$
\underbrace{\begin{bmatrix}
0 & 0 & \beta_1\left(\sum_{k=1}^m B_k^m, B_\ell^m\right) & 0 \\
0 & 0 & 0 & \beta_2\left(\sum_{i=1}^{n-1} B_i^n, B_j^n\right) \\
-\mu_2\left(\sum_{k=1}^m B_k^m, B_\ell^m\right) & 0 & 0 & \tau_2\left(\sum_{i=1}^{n-1} B_i^n, B_\ell^m\right) \\
0 & -\mu_1\left(\sum_{i=1}^{n-1} B_i^n, B_j^n\right) & -\tau_1\left(\sum_{k=1}^m B_k^m, B_j^n\right) & -\kappa_1\left(\sum_{i=1}^{n-1} B_i^n, B_j^n\right)
\end{bmatrix}}_{\mathcal{A}^N}
\underbrace{\begin{bmatrix}
\phi_k(t) \\
w_i(t) \\
\dot{\phi}_k(t) \\
\dot{w}_i(t)
\end{bmatrix}}_{y^N(t)},
$$

where $\beta_1(\cdot,\cdot), \beta_2(\cdot,\cdot), \rho_1(\cdot,\cdot), \rho_2(\cdot,\cdot), \mu_1(\cdot,\cdot), \mu_2(\cdot,\cdot), \tau_1(\cdot,\cdot), \tau_2(\cdot,\cdot)$, and $\kappa_1(\cdot,\cdot)$ refer to the sesquilinear forms shown in II.9. We represent this linear system concisely as

$$
\begin{bmatrix}
M_1^N & 0 \\
0 & M_2^N
\end{bmatrix}
\begin{bmatrix}
\dot{\vartheta}^N(t) \\
\ddot{\vartheta}^N(t)
\end{bmatrix} =
\begin{bmatrix}
0 & M_1^N \\
-A_1^N & -A_2^N
\end{bmatrix}
\begin{bmatrix}
\vartheta^N(t) \\
\dot{\vartheta}^N(t)
\end{bmatrix} \tag{IV.2}
$$

with

$$
M_1^N = \begin{bmatrix} M_{11}^N & 0 \\ 0 & M_{12}^N \end{bmatrix}, \quad
M_2^N = \begin{bmatrix} M_{21}^N & 0 \\ 0 & M_{22}^N \end{bmatrix},
$$

$$
A_1^N = \begin{bmatrix} A_{11}^N & 0 \\ 0 & A_{12}^N \end{bmatrix}, \quad
A_2^N = \begin{bmatrix} 0 & A_{31}^N \\ A_{32}^N & A_{22}^N \end{bmatrix}.
$$

The component matrices are given by

$$
\left[M_{11}^N \right]_{k,\ell} = \int_\Omega \nabla B_k^m \cdot \nabla B_\ell^m \, d\omega \qquad \left[M_{12}^N \right]_{i,j} = \int_{\Gamma_0} \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma
$$

$$
\left[M_{21}^N \right]_{k,\ell} = \int_\Omega \frac{\rho_f}{c^2} B_k^m B_\ell^m \, d\omega \qquad \left[M_{22}^N \right]_{i,j} = \int_{\Gamma_0} \rho_b B_i^n B_j^n \, d\gamma
$$

$$
\left[A_{11}^N \right]_{k,\ell} = \int_\Omega \rho_f \nabla B_k^m \cdot \nabla B_\ell^m \, d\omega \qquad \left[A_{12}^N \right]_{i,j} = \int_{\Gamma_0} EI \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma \qquad \left.\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \quad \text{(IV.3)}
$$

$$
\left[A_{31}^N \right]_{i,\ell} = -\int_{\Gamma_0} \rho_f B_\ell^m(x,0) B_i^n \, d\gamma \qquad \left[A_{32}^N \right]_{k,j} = \int_{\Gamma_0} \rho_f B_k^m(x,0) B_j^n \, d\gamma
$$

$$
\left[A_{22}^N \right]_{i,j} = \int_{\Gamma_0} c_D I \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma \,,
$$

where the finite dimensional products correspond to the infinite dimensional sesquilinear forms given in II.9.

The finite dimensional representation of Model II is similar to that given above for Model I, except the matrix A_2^N contains the additional sub-matrix A_{41}^N, which arises because of the boundary damping along Γ. For Model II, A_2^N is given by

$$
A_2^N = \left[\begin{array}{cc} A_{41}^N & A_{31}^N \\ A_{32}^N & A_{22}^N \end{array} \right] .
$$

The component matrices for Model II include all of those specified in IV.3 as well as

32

$$\left[\, A_{41}^N \,\right]_{k,\ell} = \int_\Gamma \mathsf{a}\rho_f\, B_k^m B_\ell^m \, d\gamma \tag{IV.4}$$

which corresponds to the weak form $\lambda_1(\phi, \xi)$. The linear system for Model II is given by

$$\underbrace{\begin{bmatrix} \beta_1(\sum_{k=1}^m B_k^m, B_\ell^m) & 0 & 0 & 0 \\ 0 & \beta_2(\sum_{i=1}^{n-1} B_i^n, B_j^n) & 0 & 0 \\ 0 & 0 & \rho_2(\sum_{k=1}^m B_k^m, B_\ell^m) & 0 \\ 0 & 0 & 0 & \rho_1(\sum_{i=1}^{n-1} B_i^n, B_j^n) \end{bmatrix}}_{M^N} \underbrace{\begin{bmatrix} \dot\phi_k(t) \\ \dot w_i(t) \\ \ddot\phi_k(t) \\ \ddot w_i(t) \end{bmatrix}}_{\dot y^N(t)} =$$

$$\underbrace{\begin{bmatrix} 0 & 0 & \beta_1(\sum_{k=1}^m B_k^m, B_\ell^m) & 0 \\ 0 & 0 & 0 & \beta_2(\sum_{i=1}^{n-1} B_i^n, B_j^n) \\ -\mu_2(\sum_{k=1}^m B_k^m, B_\ell^m) & 0 & \lambda_1(\sum_{k=1}^m B_k^m B_\ell^m) & \tau_2(\sum_{i=1}^{n-1} B_i^n, B_\ell^m) \\ 0 & -\mu_1(\sum_{i=1}^{n-1} B_i^n, B_j^n) & -\tau_1(\sum_{k=1}^m B_k^m, B_j^n) & -\kappa_1(\sum_i^{n-1} B_i^n, B_j^n) \end{bmatrix}}_{\mathcal{A}^N} \underbrace{\begin{bmatrix} \phi_k(t) \\ w_i(t) \\ \phi_k(t) \\ \dot w_i(t) \end{bmatrix}}_{y^N(t)}$$

As mentioned previously, the rows of each block sub-matrix are generated by holding ℓ and j fixed while varying k and i as appropriate for the particular product being computed.

The general form of matrices $M_1^N \vartheta^N(t)$ and $A_2^N \dot\vartheta^N(t)$ is presented below to help the reader better conceptualize the overall structure of the system. The block structure of M_1^N and A_2^N is characteristic of the larger matrices M^N and $\mathcal{A}^N$ for both Models I and II. Also, the products represented by $(\cdot, \cdot)$ in the matrices below correspond to those given for Model I sub-matrices $M_{11}^N, M_{12}^N, A_{22}^N, A_{31}^N$, and A_{32}^N in IV.3. We represent these matrices as

$M_1^N \vartheta^N(t) =$

33

$$
\begin{bmatrix}
(B_1^m,B_1^m) & (B_2^m,B_1^m) & \cdots & (B_m^m,B_1^m) & 0 & \cdots & 0 \\
(B_1^m,B_2^m) & \cdots & \cdots & (B_m^m,B_2^m) & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(B_1^m,B_{m-1}^m) & \cdots & \cdots & (B_m^m,B_{m-1}^m) & 0 & \cdots & 0 \\
(B_1^m,B_m^m) & \cdots & (B_{m-1}^m,B_m^m) & (B_m^m,B_m^m) & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & (B_1^n,B_1^n) & \cdots & (B_{n-1}^n,B_1^n) \\
0 & \cdots & 0 & 0 & (B_1^n,B_2^n) & \cdots & (B_{n-1}^n,B_2^n) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & (B_1^n,B_{n-2}^n) & \cdots & (B_{n-1}^n,B_{n-2}^n) \\
0 & \cdots & 0 & 0 & (B_1^n,B_{n-1}^n) & \cdots & (B_{n-1}^n,B_{n-1}^n)
\end{bmatrix}
\begin{bmatrix}
\phi_1^N(t) \\ \phi_2^N(t) \\ \vdots \\ \phi_{m-1}^N(t) \\ \phi_m^N(t) \\ w_1^N(t) \\ w_2^N(t) \\ \vdots \\ w_{n-2}^N(t) \\ w_{n-1}^N(t)
\end{bmatrix}
$$

and $A_2^N \dot\vartheta^N(t) =$

$$
\begin{bmatrix}
0 & \cdots & 0 & 0 & (B_1^n,B_1^n) & \cdots & (B_{n-1}^n,B_1^n) \\
0 & \cdots & 0 & 0 & (B_1^n,B_2^n) & \cdots & (B_{n-1}^n,B_2^n) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & (B_1^n,B_{n-2}^n) & \cdots & (B_{n-1}^n,B_{n-2}^n) \\
0 & \cdots & 0 & 0 & (B_1^n,B_{n-1}^n) & \cdots & (B_{n-1}^n,B_{n-1}^n) \\
(B_1^m,B_1^m) & (B_2^m,B_1^m) & \cdots & (B_m^m,B_1^m) & (B_1^n,B_1^n) & \cdots & (B_{n-1}^n,B_1^n) \\
(B_1^m,B_2^m) & \cdots & \cdots & (B_m^m,B_2^m) & (B_1^n,B_2^n) & \cdots & (B_{n-1}^n,B_2^n) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(B_1^m,B_{m-1}^m) & \cdots & \cdots & (B_m^m,B_{m-1}^m) & (B_1^n,B_{n-2}^n) & \cdots & (B_{n-1}^n,B_{n-2}^n) \\
(B_1^m,B_m^m) & \cdots & (B_{m-1}^m,B_m^m) & (B_m^m,B_m^m) & (B_1^n,B_{n-1}^n) & \cdots & (B_{n-1}^n,B_{n-1}^n)
\end{bmatrix}
\begin{bmatrix}
\dot\phi_1^N(t) \\ \dot\phi_2^N(t) \\ \vdots \\ \dot\phi_{m-1}^N(t) \\ \dot\phi_m^N(t) \\ \dot w_1^N(t) \\ \dot w_2^N(t) \\ \vdots \\ \dot w_{n-2}^N(t) \\ \dot w_{n-1}^N(t)
\end{bmatrix}
$$

Dimensions of the matrices and sub-matrices associated with Models I and II are:

$$M^N, \mathcal{A}^N : \quad 2(m+(n-1)) \times 2(m+(n-1))$$

$$M_1^N, M_2^N, A_1^N, A_2^N : \quad (m+(n-1)) \times (m+(n-1))$$

$$M_{11}^N, M_{21}^N, A_{11}^N, A_{41}^N : \quad m \times m$$

$$M_{12}^N, M_{22}^N, A_{12}^N, A_{22}^N : \quad (n-1) \times (n-1)$$

$$A_{31}^N : \quad m \times (n-1) \qquad A_{32}^N : \quad (n-1) \times m$$

In Chapter V, we obtain specific numerical approximations of M^N and $\mathcal{A}^N$ for Models I and II, and examine the stability of these finite systems to gain insight into the stability of the infinite dimensional systems II.8 and II.19.

34

V.   SPECIFIC APPROXIMATIONS AND RESULTS

In this chapter we select specific basis functions for discretization of the beam and cavity, discuss development of the computer programs used to generate matrices of the finite systems developed in the previous chapter, and present results for specific approximations.

Cubic splines and tensor products of Legendre polynomials are chosen as basis functions for the beam and cavity spaces, respectively (We refer to the tensor products of Legendre polynomials as "tensored Legendre polynomials" throughout this paper.). Since these choices are by no means the only possibilities, the interested reader is referred to [Ref. 9] for a discussion of alternate choices as well as selection criteria which includes: smoothness requirements, uniform preservation of exponential stability of approximating systems, accuracy, sparsity of system matrices, and ease of implementation.

The cubic splines used as a basis for H_b^n satisfy the smoothness requirements and are easily adapted to satisfy the clamped boundary conditions. We construct the set $\{B_i^n\}_{i=1}^{n-1}$ by first partitioning the beam into n uniform intervals of step size $h = \frac{b-a}{n}$. Letting $\widehat{B}_i^n$ denote the standard cubic spline corresponding to this partition, then the basis functions for the beam discretization are taken to be

$$B_1^n = \widehat{B}_0^n - 2\widehat{B}_1^n - 2\widehat{B}_{-1}^n$$

$$B_i^n = \widehat{B}_i^n \quad \text{for } i = 2, 3, \cdots, n-2$$

$$B_{n-1}^n = \widehat{B}_n^n - 2\widehat{B}_{n-1}^n - 2\widehat{B}_{n+1}^n \, ,$$

where the standard cubic splines, as defined in [Ref. 11], are given by

$$\hat{B}_i(x) = \frac{1}{h^3} \begin{cases} (x - x_{i-2})^3, & \text{if } x \in [x_{i-2}, x_{i-1}] \\ h^3 + 3h^2(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3, & \text{if } x \in [x_{i-1}, x_i] \\ h^3 + 3h^2(x_{i+1} - x) + 3h(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3, & \text{if } x \in [x_i, x_{i+1}] \\ (x_{i+2} - x)^3, & \text{if } x \in [x_{i+1}, x_{i+2}] \\ 0, & \text{otherwise}. \end{cases}$$

All of the basis functions B_i^n do in fact satisfy the clamped boundary conditions

$$B_i^n(0) = \partial_x B_i^n(0) = B_i^n(a) = \partial_x B_i^n(a) = 0$$

for $i = 1, 2, \cdots, n-1$ as can be seen in Figure 4 below where the interval $[0, 1]$ is partitioned into 10 uniform subintervals. The dashed curves represent B_1^n and B_{n-1}^n which have compact support over three intervals. The interior splines, which have compact support over four intervals, are denoted by solid lines.

Tensored Legendre polynomials are used as a basis for H_c^m. As stated in [Ref. 9], these polynomials produce smaller, more structured matrices than those obtained with linear splines or finite elements, and the natural boundary conditions along the cavity walls obviate modification of the basis elements to satisfy some essential boundary conditions. Authors of [Ref. 9] assert, however, that these benefits are not as critical in the 2-D case as they are in the 3-D problem where system matrices are considerably larger.

The basis set of tensored Legendre polynomials is obtained by forming the product of transformed Legendre polynomials, denoted $L_i^a(x)$ and $L_j^l(y)$, where the subscripts i, j indicate the degree of the polynomial, from the interval $[-1, 1]$ to $[0, a] \times [0, l]$, respectively. The transformed polynomials are obtained by substituting an appropriate linear transformation for x in the recursive definition of the standard Legendre polynomials:

$$P_{n+1}(x) = \frac{1}{n+1}[(2n+1)xP_n(x) - xP_{n-1}(x)] \qquad (V.1)$$

36
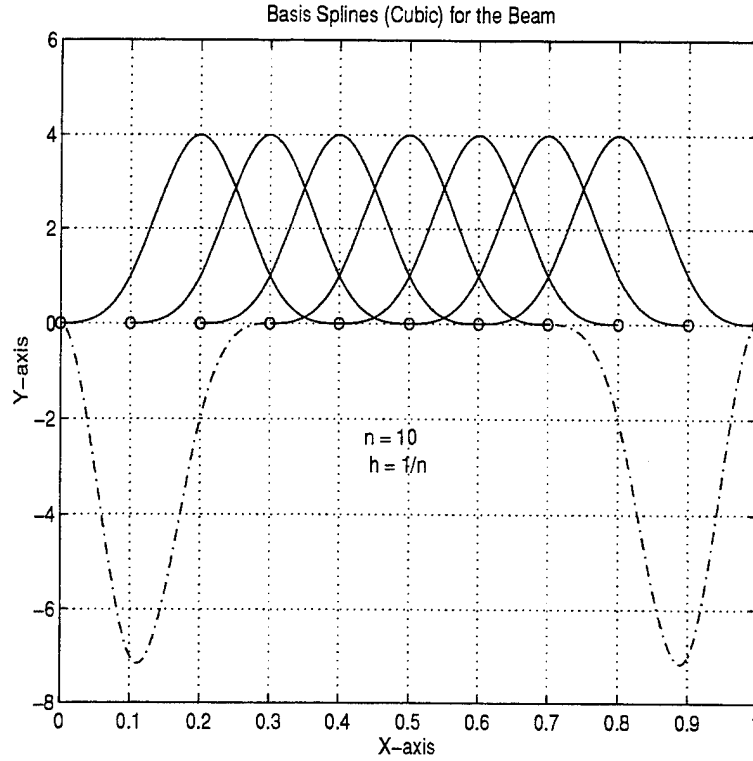
Figure 4. Cubic Polynomial Splines

with $P_0(x) = 1$, $P_1(x) = x$.

For example, substituting $\frac{t+1}{2}$ for x in definition V.1 transforms the standard Legendre polynomials from $[-1, 1]$ to $[0, 1]$. Orthogonality of the Legendre polynomials is preserved under this linear transformation (see Figure 5). Recalling that the set of basis functions for the cavity is denoted $\{B_k^m\}_{k=1}^m$, we define B_k^m as

$$B_k^m(x, y) = L_i^a(x) L_j^l(y) \quad \text{for} \quad \begin{cases} k = 1, 2, \cdots, m \\ i = 0, 1, \cdots, m_x + 1 \\ j = 0, 1, \cdots, m_y + 1 \end{cases}, \qquad (V.2)$$

where we impose the condition $i + j \neq 0$ to eliminate constant functions (i.e., exclude $L_0^a(x) L_0^l(y) = 1$ for all (x, y)) ensuring the set of functions is suitable as a basis for the quotient space. Hence, the dimension of the cavity space is $m = (m_x + 1)(m_y + 1) - 1$. For consistency throughout this paper and in our computational algorithms,

37
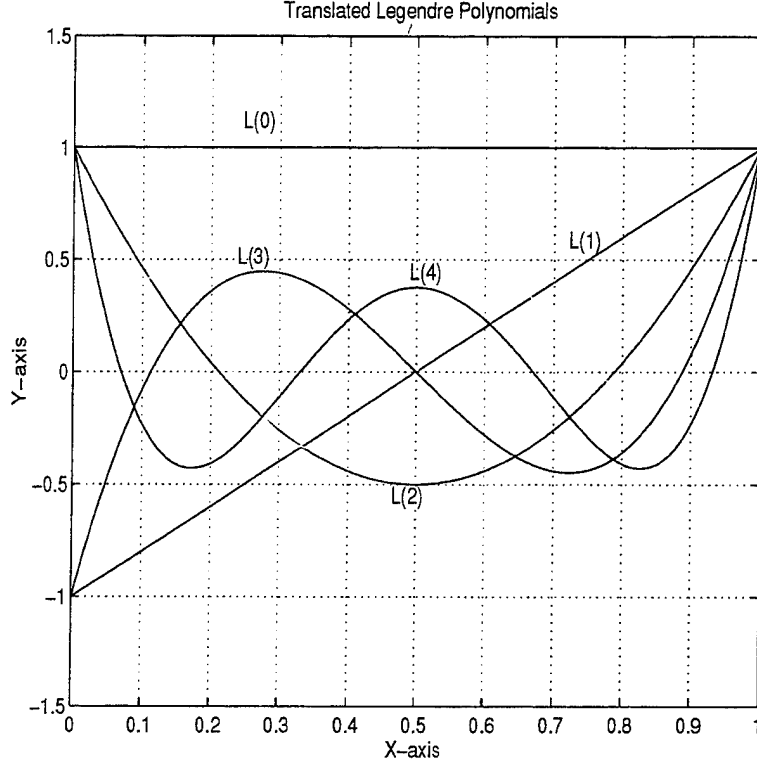
Figure 5. Transformed Legendre Polynomials

the subscript k shown in V.2 is determined by holding j fixed while i varies. As an example, the indexing scheme for $m_x = 2$ and $m_y = 2$ is

$$
\begin{aligned}
B_1^m(x,y) &= L_1^a(x)L_0^l(y) & B_5^m(x,y) &= L_2^a(x)L_1^l(y) \\
B_2^m(x,y) &= L_2^a(x)L_0^l(y) & B_6^m(x,y) &= L_0^a(x)L_2^l(y) \\
B_3^m(x,y) &= L_0^a(x)L_1^l(y) & B_7^m(x,y) &= L_1^a(x)L_2^l(y) \\
B_4^m(x,y) &= L_1^a(x)L_1^l(y) & B_8^m(x,y) &= L_2^a(x)L_2^l(y)\,.
\end{aligned}
\tag{V.3}
$$

Having selected basis functions for the beam and cavity spaces, we now turn our attention to the computation of the various component matrices associated with Models I and II. All computations are done using MATLAB (MATLAB is a high-performance interactive software package produced by *The MathWorks, Inc.*, for scientific and engineering numeric computation.). Programs written for our numerical work are found in Appendix A.

38

Matrices M_{11}^N, M_{21}^N, and A_{11}^N are all computed similarly since they each involve integration over the cavity space. In Chapter IV we defined these matrices as

$$\left[M_{11}^N \right]_{k,\ell} = \int_\Omega \nabla B_k^m \cdot \nabla B_\ell^m \, d\omega \,, \qquad \left[M_{21}^N \right]_{k,\ell} = \int_\Omega \frac{\rho_f}{c^2} B_k^m B_\ell^m \, d\omega \,,$$
$$\left[A_{11}^N \right]_{k,\ell} = \int_\Omega \rho_f \nabla B_k^m \cdot \nabla B_\ell^m \, d\omega \,. \tag{V.4}$$

The matrices are computed by a routine suggested in [Ref. 9]. For the moment, consider the integrand of M_{11}^N where $B_k^m = L_i^a(x)L_j^l(y)$ and $B_\ell^m = L_p^a(x)L_q^l(y)$:

$$\begin{aligned}
\nabla B_k^m \cdot \nabla B_\ell^m &= (\partial_x B_k^m, \partial_y B_k^m) \cdot (\partial_x B_\ell^m, \partial_y B_\ell^m) \\
&= (\partial_x(L_i^a L_j^l), \partial_y(L_i^a L_j^l)) \cdot (\partial_x(L_p^a L_q^l), \partial_y(L_p^a L_q^l)) \\
&= (L_j^l \partial_x L_i^a, L_i^a \partial_y L_j^l) \cdot (L_q^l \partial_x L_p^a, L_p^a \partial_y L_q^l) \\
&= (L_j^l L_q^l \partial_x L_i^a \partial_x L_p^a) + (L_i^a L_p^a \partial_y L_j^l \partial_y L_q^l) \,.
\end{aligned}$$

Because of the structure of the integrand, we make use of the tensor properties of the transformed Legendre basis functions to construct M_{11}^N, M_{21}^N, and A_{11}^N. Orthogonality of the transformed polynomials reduces computational complexity since

$$\int_0^l L_j^l L_q^l = 0 \quad \text{and} \quad \int_0^a L_i^a L_p^a = 0 \text{ whenever } j \ne q \text{ and } i \ne p \,.$$

First construct fundamental $(m_x + 1) \times (m_x + 1)$ matrices M_a^m and K_a^m given by

$$[M_a^m]_{ip} = \int_0^a L_i^a(x)L_p^a(x) \, dx \quad \text{and} \quad [K_a^m]_{ip} = \int_0^a \partial_x L_i^a(x)\partial_x L_p^a(x) \, dx$$

with similar definitions for M_ℓ^m and K_ℓ^m (in fact, $M_a^m = M_\ell^m$ and $K_a^m = K_\ell^m$ when $m_x = m_y$ and $[0, a] = [0, l]$). By orthogonality of the transformed Legendre polynomials, matrices M_a^m and M_ℓ^m are diagonal. The matrices $\widehat{M_{11}^N}$ and $\widehat{M_{21}^N}$ are formed by computing

$$\widehat{M_{11}^N} = M_\ell^m \otimes K_a^m + K_\ell^m \otimes M_a^m \quad \text{and} \quad \widehat{M_{21}^N} = \frac{\rho_f}{c^2} M_\ell^m \otimes M_a^m \,.$$

The symbol $\otimes$ denotes the tensor product, which we accomplish by using MATLAB's *kron* function. The ordering in the above definition is not obvious; however, attention

39

to the indexing scheme shown in V.3 and as described in Chapter IV delineate our convention. M_{11}^N and M_{21}^N are obtained by removing the first row and column of $\widehat{M}_{11}^N$ and $\widehat{M}_{21}^N$ reflecting the deletion of the constant function from the basis set. Since we take ρ_f to be constant in this paper, A_{11}^N is trivial to compute. Matrices M_{11}^N and A_{11}^N are positive definite and symmetric—although not sparse. M_{21}^N is diagonal, positive definite. The program *matten.m,* written to generate these matrices, computes the transformed Legendre polynomials and their derivatives iteratively. Integration of the differentiated transformed Legendre polynomials is accomplished by using Gaussian quadrature, while the orthogonality relation for Legendre polynomials, given by

$$\frac{b-a}{2} \int_{-1}^{1} P_r(t) P_s(t)\, dt = \frac{b-a}{2} \frac{2}{2r+1} \delta_{rs} \, ,$$

is used to compute the integrals of the translated Legendre polynomials (δ_{rs} denotes the standard Kronecker delta: $\delta_{rs} = 0$ if $r \neq s$, $= 1$ if $r = s$). Note that $\frac{b-a}{2}$ is a simple transformation factor which enables one to use this exact integration formula for integration over an interval $[a, b]$.

Since matrices $M_{12}^N, M_{22}^N, A_{12}^N,$ and A_{22}^N, given by

$$\left[M_{12}^N \right]_{i,j} = \int_{\Gamma_0} \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma \, , \qquad \left[M_{22}^N \right]_{i,j} = \int_{\Gamma_0} \rho_b B_i^n B_j^n \, d\gamma \, ,$$
$$\left[A_{12}^N \right]_{i,j} = \int_{\Gamma_0} EI \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma \, , \qquad \left[A_{22}^N \right]_{i,j} = \int_{\Gamma_0} c_D I \partial_x^2 B_i^n \partial_x^2 B_j^n \, d\gamma \, , \qquad \text{(V.5)}$$

all involve cubic spline functions or their second derivatives, they are computed similarly. The program *myspline.m* is used to generate the set of basis splines. Intrinsic MATLAB functions *polyder* and *conv* are used to differentiate and compute the product of the cubic splines. A simple program, *polyint.m* , performs the necessary integration. The programs used to compute $M_{12}^N, M_{22}^N, A_{12}^N,$ and A_{22}^N capitalize on the symmetry of the spline functions (*mat1222.m* computes M_{12}^N, A_{12}^N, and A_{22}^N; *matm22.m* computes M_{22}^N). Because the splines are equal to zero outside their regions of compact support, these four matrices become seven-banded for $n \geq 10$. All four are symmetric and positive definite in construction.

The elements of matrices A_{31}^N and A_{32}^N correspond to the integrated product of the collapsed tensored Legendre polynomials (i.e., $B_\ell^m = L_i^a(x)L_j^l(y)|_{y=0}$) and the cubic splines over the interval $[0, a]$. Recall these matrices are given by

$$\left[A_{31}^N \right]_{i,\ell} = -\int_{\Gamma_0} \rho_f B_i^n(x) B_\ell^m(x,0)\, d\gamma \quad \text{and} \quad \left[A_{32}^N \right]_{k,j} = \int_{\Gamma_0} \rho_f B_k^m(x,0) B_j^n(x)\, , d\gamma.$$

These two matrices are computed using the function *a3132.m* . Integration, done using Gaussian quadrature, is accomplished using MATLAB's intrinsic function *polyval* to evaluate polynomials at translated Gaussian knots. Note that computation of each element of A_{31}^N and A_{32}^N actually requires three or four integrations rather than just one (three if splines B_1^{n-1} or B_{n-1}^{n-1} appear in the integrand; four for integrands involving interior splines). This is due to the piecewise construction of the cubic splines over their respective regions of compact support. For example, if B_i^n is an interior spline, integration over Γ_0 is given by

$$
\begin{aligned}
\int_{\Gamma_0} \rho_f B_i^n(x) B_\ell^m(x,0)\, d\gamma &= \int_{\Gamma_0} \rho_f B_i^n(x) B_\ell^m(x,0)\, d\gamma \\
&= \int_{x-2}^{x-1} \rho_f B_i^n(x) B_\ell^m(x,0)\, dx + \int_{x-1}^{x} \rho_f B_i^n(x) B_\ell^m(x,0)\, dx \\
&\quad + \int_{x}^{x+1} \rho_f B_i^n(x) B_\ell^m(x,0)\, dx + \int_{x+1}^{x+2} \rho_f B_i^n(x) B_\ell^m(x,0)\, dx\,.
\end{aligned}
$$

Although A_{31}^N and A_{32}^N are both full matrices, their computation is simplified since each has a well-defined block structure. Further, since we take ρ_f to be constant, A_{32}^N is precisely the negative transpose of A_{31}^N (i.e., $A_{32}^N = (-A_{31}^N)^T$).

The function *mata41.m* is used to compute

$$
\begin{aligned}
\left[A_{41}^N \right]_{k,\ell} &= \int_\Gamma \mathsf{a} \rho_f B_k^m B_\ell^m\, d\gamma \\
&= \mathsf{a}\rho_f \left[\int_{\Gamma_1} B_k^m(0,y) B_\ell^m(0,y)\, dy + \int_{\Gamma_2} B_k^m(x,l) B_\ell^m(x,l)\, dx \right. \\
&\quad \left. + \int_{\Gamma_3} B_k^m(a,y) B_\ell^m(a,y)\, dy \right].
\end{aligned}
$$

Fundamental matrices corresponding to integration across Γ_1, Γ_2, and Γ_3 are computed and then summed to generate A_{41}^N. Because of the well-defined, block diagonal

41

structure of the matrices corresponding to integration across Γ_1 and Γ_3, these matrices are computed simultaneously. The matrix produced by integration across Γ_2, although not sparse, possesses symmetry along diagonals which simplifies its construction. Given the structures of these fundamental matrices, A_{41}^N is symmetric with a well-defined block-diagonal and symmetric off-diagonal construction.

In light of the structures of the component matrices discussed above, we note that for both Models I and II: (i) the matrices A_1^N and M^N are symmetric and positive definite by construction, and (ii) A_2^N has symmetric and skew symmetric block construction ($A \in \mathcal{R}^{n \times n}$ is said to be *skew symmetric* if $A^T = -A$.).

We are now ready to examine the stability of the approximation schemes developed for Models I and II. For our numerical work we assume the following parameters, which according to [Ref. 9], are physically reasonable for a $.6m$ by $1m$ cavity:

$$a = .6m\,, \qquad l = 1m\,, \qquad \rho_f = 1.21 \tfrac{kg}{m^3}$$

$$c^2 = 117649 \tfrac{m^2}{sec^2}\,, \quad \rho_b = 1.35 \tfrac{kg}{m}\,, \quad EI = 73.96\,Nm^2$$

$$c_D I = .001 \tfrac{kgm^3}{sec}$$

Note: For Model II, we take the proportionality constant $\mathsf{a} = 1$, where a appears in the boundary condition (equation II.18)

$$\partial_n \phi = -\mathsf{a}\phi_t \quad \text{for } (x,y) \in \Gamma\,, \ t > 0\,.$$

MODEL I : For $n = m_x = m_y = 6, 7, \cdots, 18$ the margins of stability for the open loop system are listed in Table I. For each n, the locations of the eigenvalues, λ, are displayed in Figures 6, 7 and 8. Eigenvalues were obtained using MATLAB's toolbox function $eig(\mathcal{A}^N, M^N)$, where matrices $\mathcal{A}^N$ and M^N are shown in equation IV.1. Eigenvalues having real parts with magnitude greater than 1 are excluded from our plots in order to better see the distribution near the imaginary axis.

For small n, the results obtained agree very favorably with those reported in [Ref. 9]. Comparison of the values shown in Table I indicates that there does not appear to be a uniform margin of stability between the eigenvalues and the imaginary axis (i.e., the data in Table I does not indicate that the maximum $\Re(\lambda)$ for the cases tested is converging to a limit.). This is what we expect based on the conclusions contained in [Ref. 5]; however, we are somewhat hesitant to report this finding since positive eigenvalues appear in our results for $n \geq 17$. These positive eigenvalues should not appear since Model I is dissipative, and therefore, all eigenvalues of the system should lie in the left–half complex plane (i.e., $\Re(\lambda) \leq 0$). The absence of a clear margin of stability, as well as the appearance of positive eigenvalues, may represent a numerical/computational instability problem. We offer two reasons for our suspicions.

• During the development of the programs used to compute the component matrices $M^N, \mathcal{A}^N$, we were able to delay the appearance of positive eigenvalues by incorporating more stable computational methods. Our early programs relied heavily on MATLAB's intrinsic "*poly*–type" functions (e.g., *polyder, conv, polyval*) and our simple polynomial integration program *polyint.m*. We modified our code so that integrations involving tensored Legendre polynomials—or derivatives thereof—is done either by using Gaussian quadrature or by well–known properties of the Legendre polynomials. Before these changes, we observed positive eigenvalues for $n = 13$. However, after incorporating these more stable techniques, positive eigenvalues did not appear until $n = 17$.

- We first attempted to find eigenvalues by calculating $D = (M^N)^{-1}\mathcal{A}^N$ and by using MATLAB's $eig(D)$. However, for values of $n \geq 13$, MATLAB returned warnings that D was near singular. We then sought to solve the generalized eigenvalue problem using MATLAB's $eig(\mathcal{A}^N, M^N)$, seeking a more computationally reliable algorithm for the problem at hand. Note that for $n \leq 12$, $eig(D)$ and $eig(A^N, M^N)$ returned very similar results. The consistency of the patterns shown in Figures 6, 7, and 8 (all generated using $eig(\cdot, \cdot)$) lend confidence to our belief that $eig(\mathcal{A}^N, M^N)$ provides more reliable results than does $eig(D)$ (Note that $eig(\cdot, \cdot)$ did not return any "near singular" warnings even when tested using very poorly conditioned matrices.). Nonetheless, computational instability may increase as n does since MATLAB's $eig(\cdot, \cdot)$ function uses a QZ algorithm and, according to [Ref. 14], some QZ algorithms destroy both the symmetry and positive definiteness of the semi-definite pair $(\mathcal{A}^N, M^N)$.

Finally, inspection of the eigenvalue plots appearing in Figures 6, 7 and 8 reveals a consistency in pattern even for $n = 17, 18$, when positive eigenvalues appear. Thus our computational approach does not fail catastrophically for a particular (large) n; rather, it degenerates as the matrix systems become increasingly singular as n increases.

MODEL II : Results obtained for $n = m_x = m_y = 5, 6, \cdots, 20$ are listed in Table II. Eigenvalue plots are displayed in Figures 9 and 10. The MATLAB toolbox function $eig(\mathcal{A}^N, M^N)$ is again used. Eigenvalues having real parts with magnitude greater than 30 are not displayed in order to better see the distribution near the imaginary axis.

The data reveals that while eigenvalues lie further away from the imaginary axis, as expected, given the dissipative boundary conditions assumed along Γ as well as Γ_0, no definitive uniform margin of stability appears to exist. That is, the values shown in Table II are creeping towards the imaginary axis as n increases. Although this movement cannot be seen from Figures 9 and 10, the figures do reveal consistency in the eigenvalue plots as n increases. Nonetheless, this creeping phenomena may not be an indication that the infinite dimensional system lacks uniform exponential stability. Rather, the problem may be related to our numerical/computational approach for the reasons stated above.

For Model II, we see that (i) the maximum real part of the (non-zero) eigenvalues lie further away from the imaginary axis in this model than they do in Model I, and (ii) the dimension of the approximating solution spaces can be increased without the appearance of positive eigenvalues, (at least up to $n = m_x = m_y = 20$—the extent of our testing). Thus Model II is likely the better choice for use in formulating the noise control problem.

45

$n = m_x = m_y$	$\max\{\Re(\lambda)\}$
6	-0.021207
7	-0.019998
8	-0.015566
9	-0.010471
10	-0.006223
11	-0.004555
12	-0.005533
13	-0.005440
14	-0.006347
15	-0.006806
16	-0.001130
17	+0.050618
18	+0.051744

Table I. **Model I:** Margin between the open loop eigenvalues (λ) and the imaginary axis.

$n = m_x = m_y$	$\max\{\Re(\lambda)\}$
5	-1.218388
6	-1.250145
7	-1.133252
8	-1.388610
9	-1.104220
10	-1.108291
11	-1.088381
12	-1.087316
13	-1.076611
14	-1.072313
15	-1.065708
16	-1.060883
17	-1.055837
18	-1.051932
19	-1.047767
20	-1.044687

Table II. **Model II:** Margin between the open loop eigenvalues (λ) and the imaginary axis.
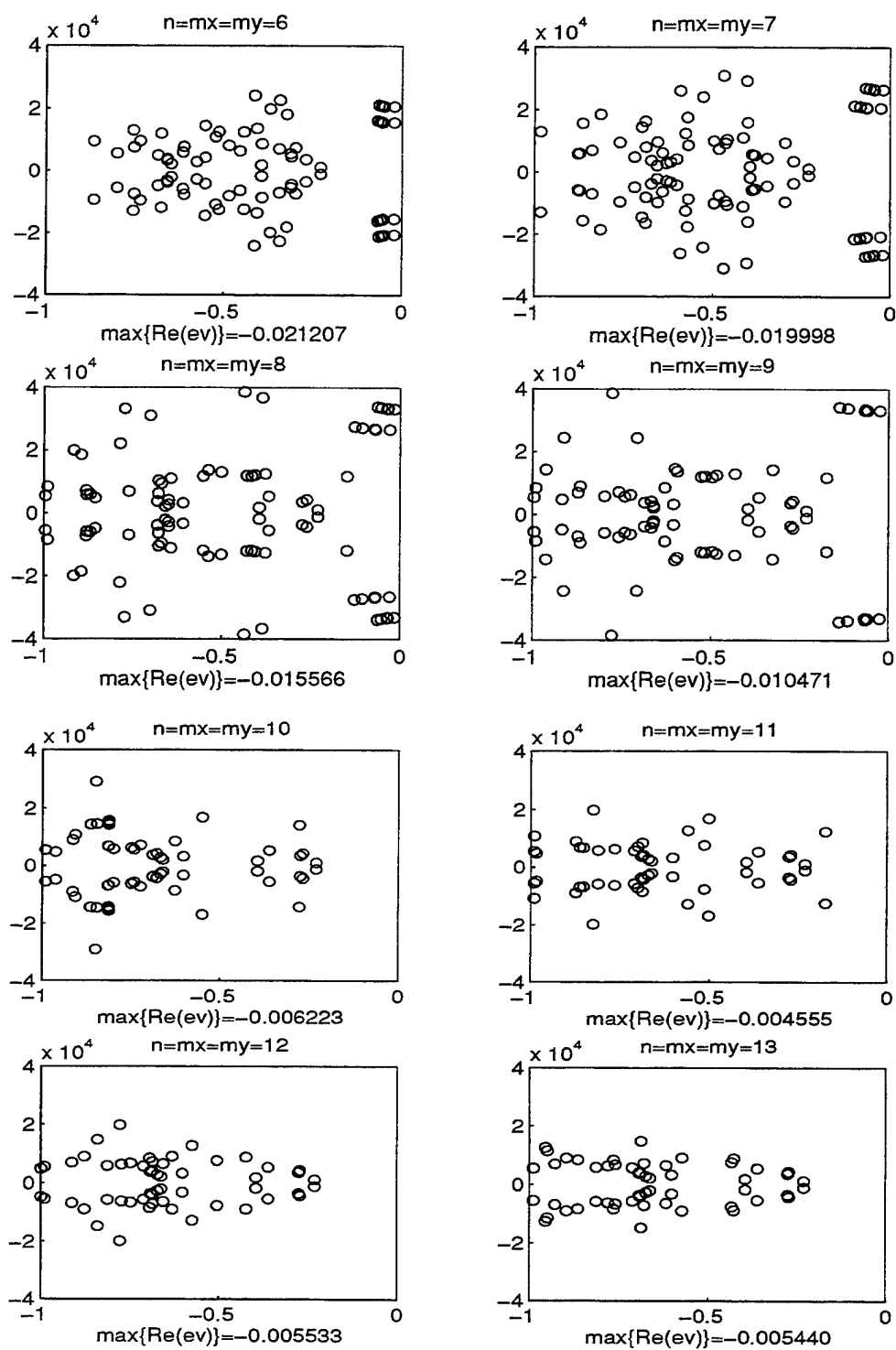
46

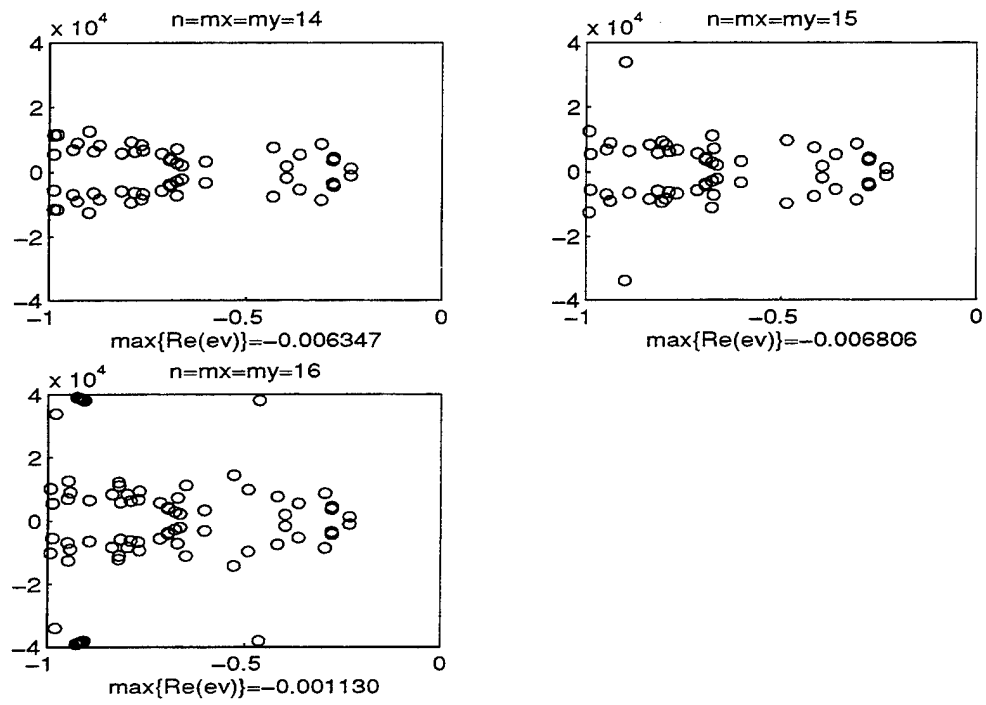Figure 6. Mod I: Eigenvalues for $n = m_x = m_y = 6, 7, 8, 9, 10, 11, 12, 13$.

Figure 7. Mod I: Eigenvalues for $n = m_x = m_y = 14, 15, 16$.
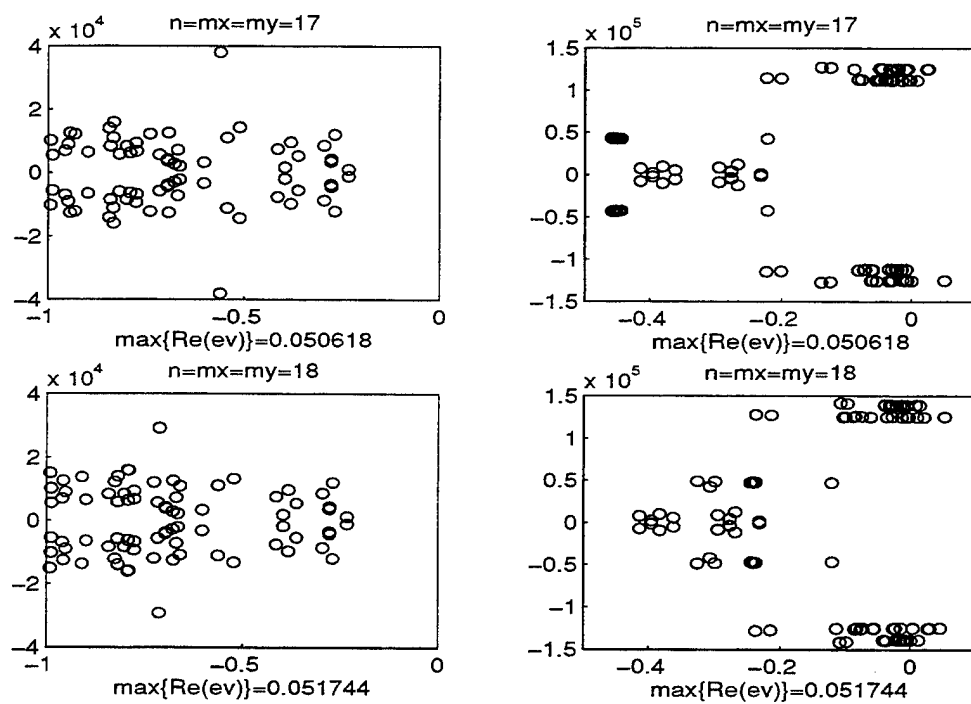
Figure 8. Mod I: Eigenvalues for $n = m_x = m_y = 17, 18$.
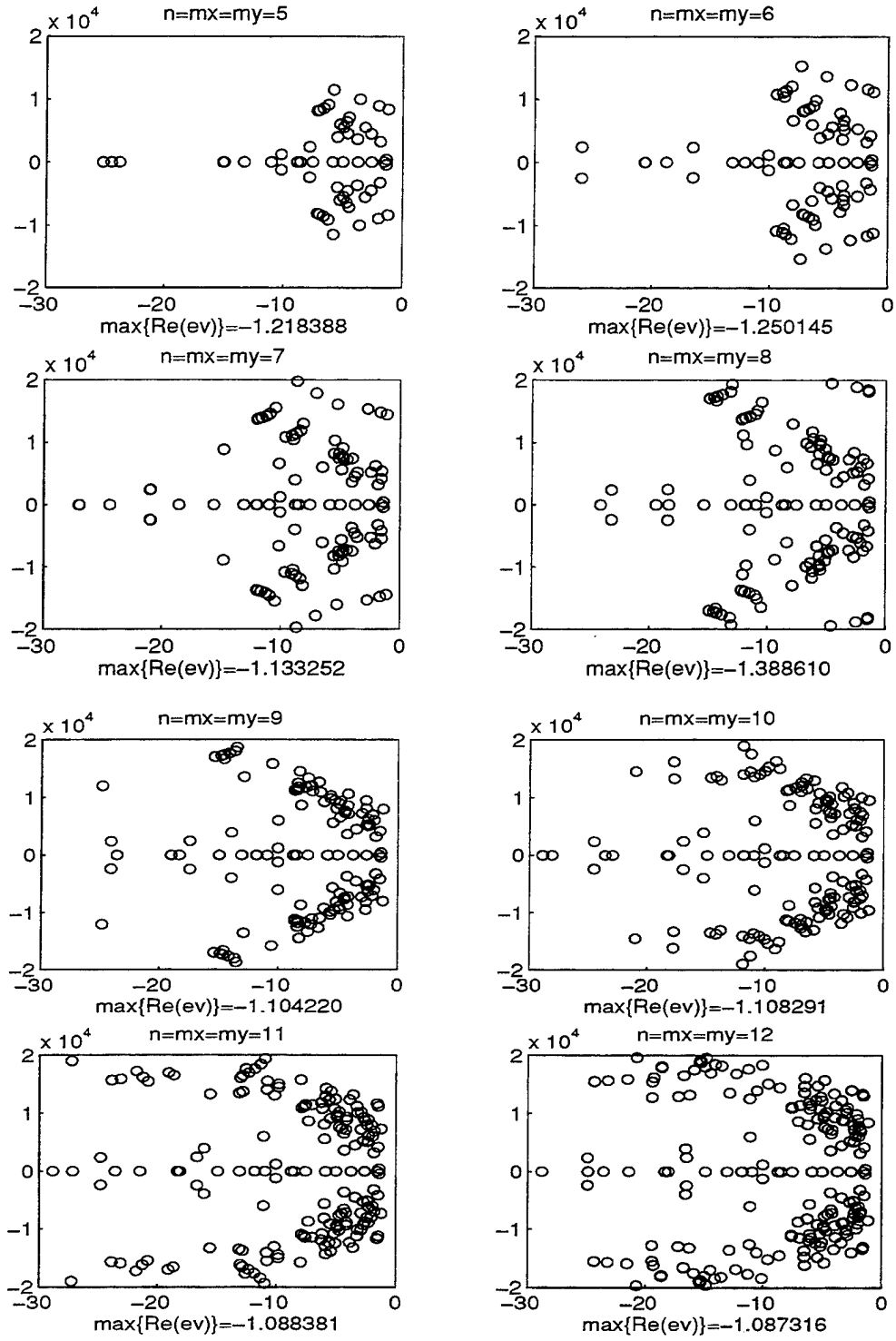
Figure 9. Mod II: Eigenvalues for $n = m_x = m_y = 5, 6, 7, 8, 9, 10, 11, 12$.
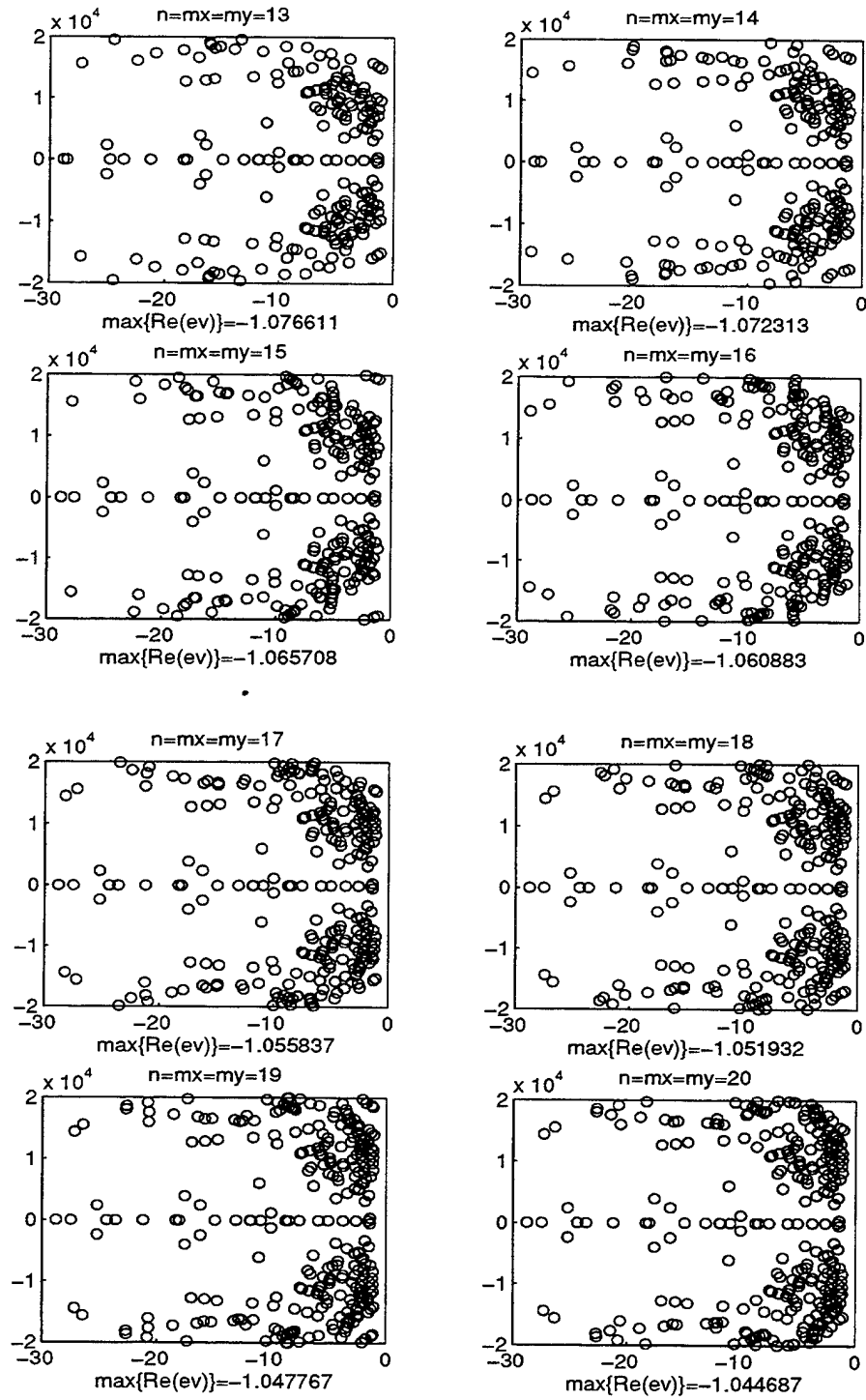
Figure 10. Mod II: Eigenvalues for $n = m_x = m_y = 13, 14, 15, 16, 17, 18, 19, 20$.

VI.   CONCLUSIONS

In this paper we investigate, by numerical approximation, the uniform exponential stability of two infinite dimensional systems developed to model the acoustic/structure interaction of a fluid-filled, rectangular cavity (known to be dissipative). Model I assumes dissipative boundary conditions along one side of the boundary, while Model II assumes dissipation boundary conditions along all four sides of the cavity. We formally obtain weak variational formulations for these two models, express each as a finite dimensional system by discretizing the solution spaces for the acoustic pressure $\phi(t, x, y)$ and transverse displacement of the beam $w(t, x)$, and use the Galerkin technique to transform the systems of PDEs into systems of ODEs. We evaluate the uniform exponential stability of these systems by examining the location of their eigenvalues in the complex plane. Eigenvalues of these systems are determined by solving the generalized eigenvalue problem $(\lambda M^N - \mathcal{A}^N)\vec{y}^N = \vec{0}$. We found that:

- The numerical approximations do not reflect the existence of uniform margins of stability for either model. The maximum real eigenvalues do not appear to be converging towards a greatest upper bound as the dimensions of the finite systems increase. Nonetheless, our numerical results clearly indicate that Model II provides a wider margin of stability than does Model I and, thus, is likely a better choice when formulating the noise control problem.

- The choice of cubic spline and tensored Legendre polynomials—in concert with the use of the Galerkin method—(i) simplifies computation of the component matrices of M^N and $\mathcal{A}^N$, and (ii) contributes to the overall structure of M^N and $\mathcal{A}^N$ simplifying the computation of eigenvalues.

Possibilities for future work to include:

- Investigate alternate methods of solving the generalized eigenvalue problem rather than using MATLAB's *eig* $(\mathcal{A}^N, M^N)$. Alternate methods should attempt to capitalize on the structure of the semi-definite pair $(\mathcal{A}^N, M^N)$. MATLAB's *eig* $(\cdot, \cdot)$ function uses the QZ algorithm and may be destroying both the symmetry and positive definiteness of the pair (A^N, M^N) as some QZ algorithms do. An alternate approach to the generalized eigenvalue problem is suggested in [Ref. 14].

- Assume different dissipative boundary conditions and numerically analyze the stability of these systems using various approximation schemes. Consider models with medium damping and/or different coupling mechanisms between the acoustic and the structure components [Ref. 5].

- Investigate the numerical stability and preservation of exponential stability of the approximation schemes presented in this paper with different choices of basis functions to discretize the beam and cavity solution spaces, or use different schemes altogether.

- Investigate other mathematical libraries such as NAG or IMSL, which may have reliable subroutines for solving the generalized eigenvalue problem presented in this paper.

APPENDIX. MATLAB FUNCTION AND SCRIPT FILES

This appendix contains the programs used to compute the eigenvalues and produce the eigenvalue plots shown in this paper.

1. Below are examples of the MATLAB *script* files which call the various *function* files shown in this appendix, as well as several intrinsic MATLAB files, for eigenvalue computation and plotting.

```
****************    Model I, Eigenvalue Computation    *************
n=6; mx=n; my=n; a=0; b=.6; el=1; rhof=1.21; rhob=1.35; c=sqrt(117649);
EI=73.96; cdI=.001;  [M11,M21,A11]=matten(mx,my,b,el,rhof,c);
[M12,A12,A22]=mat1222(a,b,n,EI,cdI);  [M22]=matm22(a,b,n,rhob);
[A31,A32]=a3132(a,b,n,mx,my,rhof); m=(mx+1)*(my+1)-1; nm1=n-1; t=m+nm1;
T=zeros(t,t);  M1=[M11 zeros(m,nm1); zeros(nm1,m) M12];
M2=[M21 zeros(m,nm1); zeros(nm1,m) M22]; M=[M1 T; T M2];
A1=[-A11 zeros(m,nm1); zeros(nm1,m) -A12];
A2=[zeros(m,m) -A31; -A32 -A22];  A=[T M1; A1 A2];
ev6=eig(full(A),full(M));  e6=max(real(ev6)); subplot(2,2,1); tt=4*10^4;
axis([-1 0 -tt tt]); hold; w=[0 0]; q=[-tt tt]; plot(w,q);
ww=[-1 0]; qq=[tt tt]; plot(ww,qq);  plot(ev6,'o');  title('n=mx=my=6')

****************    Model II, Eigenvalue Computation    **************
n=6; mx=n; my=n; a=0; b=.6; el=1; rhof=1.21; rhob=1.35; c=sqrt(117649);
EI=73.96; cdI=.001;  [M11,M21,A11]=matten(mx,my,b,el,rhof,c);
[M12,A12,A22]=mat1222(a,b,n,EI,cdI);  [M22]=matm22(a,b,n,rhob);
[A31,A32]=a3132(a,b,n,mx,my,rhof);  A41]=a41(b,el,mx,my,rhof,c);
m=(mx+1)*(my+1)-1;  nm1=n-1;  t=m+nm1;  T=zeros(t,t);
M1=[M11 zeros(m,nm1); zeros(nm1,m) M12];
M2=[M21 zeros(m,nm1); zeros(nm1,m) M22];  M=[M1 T; T M2];
A1=[-A11 zeros(m,nm1); zeros(nm1,m) -A12];  A2=[A41 -A31; -A32 -A22];
A=[T M1; A1 A2];  ev6=eig(full(A),full(M));  e6=max(real(ev6));
subplot(2,2,2); tt=2*10^4; axis([-30 0 -tt tt]); hold;  w=[0 0];
q=[-tt tt];  plot(w,q);  ww=[-30 0];  qq=[tt tt];   plot(ww,qq);
plot(ev6,'o');   title('n=mx=my=6')
*******************************************************************
```

2. Function file *matten.m* computes M_{11}^N, M_{21}^N and A_{11}^N.


```
****************************************************************************
function [M11,M21,A11]=matten(mx,my,b,el,rhof,c)

% [M11 M21 A11] = matten(mx,my,b,el,rhof,c)
%
% This function produces matrices M11, M21, and A11--all are
%       (mx+1)*(my+1)-1 by (mx+1)*(my+1)-1
% Input:  mx = highest degree of Legendre basis poly for x-axis
%    my = highest degree of Legendre basis poly for y-axis
%    b = right end point along x-axis (i.e., [0,b])
%    el = right end point along y-axis (i.e., [0,el])
%    rhof = uniform density of fluid
%    c = speed of acoustic wave in fluid
% Written by Major J. M. Shehan, last update 21 May 95.

% Begin matten.m

%%%  Compute M11
if mx == my & b == el
  mx1=mx+1; x=ones(1,mx1); vx=1:2:2*mx1; intPPx=b*(x./vx); Ma=intPPx;
  % Determine Gaussian weights (w(i)) & evaluation points (x(i)).
    x=1:1:mx-1;  x=x./sqrt((2*x+1).*(2*x-1));  j=diag(x,1)+diag(x,-1);
    [u x]=eig(j);  x=diag(x);  [x i]=sort(x);  u=u(:,i);  w=u(1,:).^2;
    w=w'.*2;  dx=b/2;
  for i=1:mx; s=x(i); p(i,1)=1; p(i,2)=s;
      dpn(i,1)=0; dpn(i,2)=1;
    for j=2:mx
      p(i,j+1)=((2*j-1)*s*p(i,j)-(j-1)*p(i,j-1))/j;
      dpn(i,j+1)=((2*j-1)*s*dpn(i,j)-(j-1)*dpn(i,j-1)+(2*j-1)*p(i,j))/j;
    end
  end; DPxval=dpn';
  for i=1:mx1
    for j=1:mx1
      Ka(i,j)=(2/b)*sum(w'.*DPxval(i,:).*DPxval(j,:));
    end
  end;
  %%%%% Mel=Ma and Kel=Ka when b=el and mx=my.
  Mad=diag(Ma);  sMad=sparse(Mad); sKa=sparse(Ka);
  M11T=kron(sMad,sKa)+kron(sKa,sMad); [row,col]=size(M11T);
```

56

```
     M11=M11T(2:row,2:col); A11=rhof*sparse(M11); %M11=full(M11);
     %%%  Compute M21
     M21T=kron(Ma,Ma);  t=length(M21T);
     M21=sparse(diag((rhof/c^2)*M21T(2:t)));
  else
   %%% Compute integrals of Legendre poly's.
    mx1=mx+1; x=ones(1,mx1); vx=1:2:2*mx1; intPPx=b*(x./vx); Ma=intPPx;
    Mad=diag(Ma);
    my1=my+1; y=ones(1,my1); vy=1:2:2*my1; intPPy=el*(y./vy); Mel=intPPy;
    Meld=diag(Mel);
  % Compute deriv's & eval 'product' integrals of translated Legendre's.
    % For x-axis: Determine Gauss weights (w(i)) & eval points (x(i)).
      x=1:1:mx-1;  x=x./sqrt((2*x+1).*(2*x-1));  j=diag(x,1)+diag(x,-1);
      [u x]=eig(j);  x=diag(x);  [x i]=sort(x);  u=u(:,i);  w=u(1,:).^2;
      w=w.*2;  dx=b/2;
    for i=1:mx; s=x(i); p(i,1)=1; p(i,2)=s;
        dpn(i,1)=0; dpn(i,2)=1;
      for j=2:mx
        p(i,j+1)=((2*j-1)*s*p(i,j)-(j-1)*p(i,j-1))/j;
        dpn(i,j+1)=((2*j-1)*s*dpn(i,j)-(j-1)*dpn(i,j-1)+(2*j-1)*p(i,j))/j;
      end
    end; DPxval=dpn';
    for i=1:mx1
      for j=1:mx1
        Ka(i,j)=(2/b)*sum(w.*DPxval(i,:).*DPxval(j,:));
      end
    end
  % For y-axis: Determine Gauss weights (w(i)) & eval points (y(i)).
      y=1:1:my-1;  y=y./sqrt((2*y+1).*(2*y-1));  j=diag(y,1)+diag(y,-1);
      [u y]=eig(j);  y=diag(y);  [y i]=sort(y);  u=u(:,i);
      wy=u(1,:).^2; wy=wy.*2;  dy=el/2;
    for i=1:my; sy=y(i); py(i,1)=1; py(i,2)=sy;
        dpny(i,1)=0; dpny(i,2)=1;
      for j=2:my
        py(i,j+1)=((2*j-1)*sy*py(i,j)-(j-1)*py(i,j-1))/j;
  dpny(i,j+1)=((2*j-1)*sy*dpny(i,j)-(j-1)*dpny(i,j-1)+(2*j-1)*py(i,j))/j;
      end
    end; DPyval=dpny';
    for i=1:my1
      for j=1:my1
        Kel(i,j)=(2/el)*sum(wy.*DPyval(i,:).*DPyval(j,:));
      end
```

```
  end
    %%% Compute M11 & A11
    sMad=sparse(Mad);   sKa=sparse(Ka); sMeld=sparse(Meld);
    sKel=sparse(Kel);
    M11T=kron(sMeld,sKa)+kron(sKel,sMad);   [row,col]=size(M11T);
    M11=M11T(2:row,2:col); A11=rhof*M11; % M11=full(M11);
    %%%  Compute M21
    M21T=kron(Mel,Ma); t=length(M21T);
    M21=sparse(diag((rhof/c^2)*M21T(2:t)));
end  % End 'if' statement.
end  % End matten.m
**************************************************************************
```

3. Function file *mat1222.m* computes M_{12}^N, A_{12}^N, and A_{22}^N.

```
****************************************************************************
function [M12,A12,A22] = mat1222(a,b,n,EI,cdI)

%   [M12 A12 A22] = mat1222(a,b,n,EI,cdI)
%
%   Returns M12, A12, and A22 matrices.
%   Input:  [a,b] = domain;
%           n = no. of symmetric partitions of interval [a,b];
%           EI = stiffness coefficient;
%           cdI = damping coefficient.
%   Note: n >= 4 required.
%   Extrinsic functions called:  myspline.m
%
%   Written by Major J. M. Shehan, updated 11 April 95.

% Begin mat1222.m
% Compute step size 'h' and generate 'x' vector.
   h=(b-a)/n;  x=a:h:b;
% Compute the cubic spline basis set for the beam.
    [B,B1,Bnm1]=myspline(a,b,n);  b1=B1(1,:); b2=B1(2,:); b3=B1(3,:);
bnm11=Bnm1(1,:); bnm12=Bnm1(2,:); bnm13=Bnm1(3,:);
% Compute 2d derivative of cubic splines.
    ddb1=polyder(polyder(b1));  ddb2=polyder(polyder(b2));
    ddb3=polyder(polyder(b3));  ddbnm11=polyder(polyder(bnm11));
    ddbnm12=polyder(polyder(bnm12));  ddbnm13=polyder(polyder(bnm13));
      [uu vv]=size(B);
    for i=13:uu-12   % i=13 is index of B(2(1))
      D2B(i-12,:)=polyder(polyder(B(i,:)));
    end
%%%  Compute M12 matrix
if n < 4
     error('n >= 4 required')
elseif n==4
   m111=polyint(conv(ddb1,ddb1),x(1),x(2));
   m112=polyint(conv(ddb2,ddb2),x(2),x(3));
   m113=polyint(conv(ddb3,ddb3),x(3),x(4));  m11=m111+m112+m113;

   m121=polyint(conv(ddb1,D2B(1,:)),x(1),x(2));
   m122=polyint(conv(ddb2,D2B(2,:)),x(2),x(3));
```

```
    m123=polyint(conv(ddb3,D2B(3,:)),x(3),x(4));  m12=m121+m122+m123;

    m131=polyint(conv(ddb2,ddbnm11),x(2),x(3));
    m132=polyint(conv(ddb3,ddbnm12),x(3),x(4));   m13=m131+m132;

    m221=polyint(conv(D2B(1,:),D2B(1,:)),x(1),x(2));
    m222=polyint(conv(D2B(2,:),D2B(2,:)),x(2),x(3));
    m223=polyint(conv(D2B(3,:),D2B(3,:)),x(3),x(4));
    m224=polyint(conv(D2B(4,:),D2B(4,:)),x(4),x(5));
    m22=m221+m222+m223+m224;

    M12=[m11 m12 m13; m12 m22 m12; m13 m12 m11];
elseif n==5
    m111=polyint(conv(ddb1,ddb1),x(1),x(2));
    m112=polyint(conv(ddb2,ddb2),x(2),x(3));
    m113=polyint(conv(ddb3,ddb3),x(3),x(4));   m11=m111+m112+m113;

    m121=polyint(conv(ddb1,D2B(1,:)),x(1),x(2));
    m122=polyint(conv(ddb2,D2B(2,:)),x(2),x(3));
    m123=polyint(conv(ddb3,D2B(3,:)),x(3),x(4));   m12=m121+m122+m123;

    m131=polyint(conv(ddb2,D2B(5,:)),x(2),x(3));
    m132=polyint(conv(ddb3,D2B(6,:)),x(3),x(4));   m13=m131+m132;

    m14=polyint(conv(ddb3,ddbnm11),x(3),x(4));

    m221=polyint(conv(D2B(1,:),D2B(1,:)),x(1),x(2));
    m222=polyint(conv(D2B(2,:),D2B(2,:)),x(2),x(3));
    m223=polyint(conv(D2B(3,:),D2B(3,:)),x(3),x(4));
    m224=polyint(conv(D2B(4,:),D2B(4,:)),x(4),x(5));
    m22=m221+m222+m223+m224;

    m231=polyint(conv(D2B(2,:),D2B(5,:)),x(2),x(3));
    m232=polyint(conv(D2B(3,:),D2B(6,:)),x(3),x(4));
    m233=polyint(conv(D2B(4,:),D2B(7,:)),x(4),x(5));
    m23=m231+m232+m233;

    m241=polyint(conv(D2B(3,:),ddbnm11),x(3),x(4));
    m242=polyint(conv(D2B(4,:),ddbnm12),x(4),x(5));   m24=m241+m242;

    m331=polyint(conv(D2B(5,:),D2B(5,:)),x(2),x(3));
    m332=polyint(conv(D2B(6,:),D2B(6,:)),x(3),x(4));
```

```
    m333=polyint(conv(D2B(7,:),D2B(7,:)),x(4),x(5));
    m334=polyint(conv(D2B(8,:),D2B(8,:)),x(5),x(6));
    m33=m331+m332+m333+m334;


    m341=polyint(conv(D2B(6,:),ddbnm11),x(3),x(4));
    m342=polyint(conv(D2B(7,:),ddbnm12),x(4),x(5));
    m343=polyint(conv(D2B(8,:),ddbnm13),x(5),x(6));
    m34=m341+m342+m343;


    m441=polyint(conv(ddbnm11,ddbnm11),x(3),x(4));
    m442=polyint(conv(ddbnm12,ddbnm12),x(4),x(5));
    m443=polyint(conv(ddbnm13,ddbnm13),x(5),x(6));
    m44=m441+m442+m443;
M12=[m11 m12 m13 m14;m12 m22 m23 m24; m13 m23 m33 m34;m14 m24 m34 m44];

else
    m111=polyint(conv(ddb1,ddb1),x(1),x(2));
    m112=polyint(conv(ddb2,ddb2),x(2),x(3));
    m113=polyint(conv(ddb3,ddb3),x(3),x(4));   m11=m111+m112+m113;

    m121=polyint(conv(ddb1,D2B(1,:)),x(1),x(2));
    m122=polyint(conv(ddb2,D2B(2,:)),x(2),x(3));
    m123=polyint(conv(ddb3,D2B(3,:)),x(3),x(4));   m12=m121+m122+m123;

    m131=polyint(conv(ddb2,D2B(5,:)),x(2),x(3));
    m132=polyint(conv(ddb3,D2B(6,:)),x(3),x(4));   m13=m131+m132;

    m14=polyint(conv(ddb3,D2B(9,:)),x(3),x(4));

    m221=polyint(conv(D2B(1,:),D2B(1,:)),x(1),x(2));
    m222=polyint(conv(D2B(2,:),D2B(2,:)),x(2),x(3));
    m223=polyint(conv(D2B(3,:),D2B(3,:)),x(3),x(4));
    m224=polyint(conv(D2B(4,:),D2B(4,:)),x(4),x(5));
    m22=m221+m222+m223+m224;

    m231=polyint(conv(D2B(2,:),D2B(5,:)),x(2),x(3));
    m232=polyint(conv(D2B(3,:),D2B(6,:)),x(3),x(4));
    m233=polyint(conv(D2B(4,:),D2B(7,:)),x(4),x(5));
    m23=m231+m232+m233;

    m241=polyint(conv(D2B(3,:),D2B(9,:)),x(3),x(4));
    m242=polyint(conv(D2B(4,:),D2B(10,:)),x(4),x(5));   m24=m241+m242;
```

```matlab
if n==6
   m25=0;  % Since D2B(13,:) does not exitst as defined above for n=6.
else
   m25=polyint(conv(D2B(4,:),D2B(13,:)),x(4),x(5));
end
% Build M12 matrix:
w1=[m11 m22*ones(1,n-3) m11]; w2=[m12 m23*ones(1,n-4) m12];
w3=[m13 m24*ones(1,n-5) m13]; w4=[m14 m25*ones(1,n-6) m14];
M121=sparse(diag(w1)+diag(w2,1)+diag(w3,2)+diag(w4,3)+diag(w2,-1));
M122=sparse(diag(w3,-2)+diag(w4,-3));
M12=M121+M122;
end
A12=EI*M12;  %  Compute A12
A22=cdI*M12;  %  Compute A22

end  % End mat1222.m
*****************************************************************************
```

4. Function file *matm22.m* computes M_{22}^N.


```
************************************************************************
function [M22] = matm22(a,b,n,rhob)

%   [M22]=matm22(a,b,n,rhob)
%
%   The function produces the (n-1)x(n-1) M22 matrix whose elements
%   are the integrals of rhob*(B(i)*B(j)) evaluated over the appro-
%   priate partitions of [a,b] where i,j=1,2,...,n-1.  B denotes
%   cubic splines.
%
%   Input: a & b = boundary of beam, [a,b];
%     n = number of symmetric partitions of interval [a,b];
%     rhob = uniform mass density of beam
%   NOTE: n must be >= 4 for this function.
%
%   Extrinsic functions called:  myspline.m
%
%   Written by Major J. M. Shehan, updated 8 April 95.


%   Begin matm22.m

% Determine step size and build x vector.
    h=(b-a)/n;    x=a:h:b;
% Compute cubic basis splines for beam; B1=B(1) & Bnm1=B(n-1).
    [B,B1,Bnm1]=myspline(a,b,n); b1=B1(1,:); b2=B1(2,:); b3=B1(3,:);
% Determine if 'n' is large enough and compute M22 matrix.
if n < 4
     error('n >= 4 required')
elseif n==4
   m111=polyint(conv(b1,b1),x(1),x(2));
   m112=polyint(conv(b2,b2),x(2),x(3));
   m113=polyint(conv(b3,b3),x(3),x(4));  m11=m111+m112+m113;

   m121=polyint(conv(b1,B(13,:)),x(1),x(2));
   m122=polyint(conv(b2,B(14,:)),x(2),x(3));
   m123=polyint(conv(b3,B(15,:)),x(3),x(4));  m12=m121+m122+m123;

   m131=polyint(conv(b2,Bnm1(1,:)),x(2),x(3));
   m132=polyint(conv(b3,Bnm1(2,:)),x(3),x(4));  m13=m131+m132;
```

```
      m221=polyint(conv(B(13,:),B(13,:)),x(1),x(2));
      m222=polyint(conv(B(14,:),B(14,:)),x(2),x(3));
      m223=polyint(conv(B(15,:),B(15,:)),x(3),x(4));
      m224=polyint(conv(B(16,:),B(16,:)),x(4),x(5));
      m22=m221+m222+m223+m224;

      M22=rhob*[m11 m12 m13; m12 m22 m12; m13 m12 m11]
 elseif n==5
      m111=polyint(conv(b1,b1),x(1),x(2));
      m112=polyint(conv(b2,b2),x(2),x(3));
      m113=polyint(conv(b3,b3),x(3),x(4));  m11=m111+m112+m113;

      m121=polyint(conv(b1,B(13,:)),x(1),x(2));
      m122=polyint(conv(b2,B(14,:)),x(2),x(3));
      m123=polyint(conv(b3,B(15,:)),x(3),x(4));  m12=m121+m122+m123;

      m131=polyint(conv(b2,B(17,:)),x(2),x(3));
      m132=polyint(conv(b3,B(18,:)),x(3),x(4));  m13=m131+m132;

      m14=polyint(conv(b3,Bnm1(1,:)),x(3),x(4));

      m221=polyint(conv(B(13,:),B(13,:)),x(1),x(2));
      m222=polyint(conv(B(14,:),B(14,:)),x(2),x(3));
      m223=polyint(conv(B(15,:),B(15,:)),x(3),x(4));
      m224=polyint(conv(B(16,:),B(16,:)),x(4),x(5));
      m22=m221+m222+m223+m224;

      m231=polyint(conv(B(14,:),B(17,:)),x(2),x(3));
      m232=polyint(conv(B(15,:),B(18,:)),x(3),x(4));
      m233=polyint(conv(B(16,:),B(19,:)),x(4),x(5));
      m23=m231+m232+m233;

      m241=polyint(conv(B(15,:),Bnm1(1,:)),x(3),x(4));
      m242=polyint(conv(B(16,:),Bnm1(2,:)),x(4),x(5));  m24=m241+m242;
M22=rhob*[m11 m12 m13 m14;m12 m22 m23 m24;m13 m23 m22 m12;m14 m24 m12 m11];

 else
      m111=polyint(conv(b1,b1),x(1),x(2));
      m112=polyint(conv(b2,b2),x(2),x(3));
      m113=polyint(conv(b3,b3),x(3),x(4));  m11=m111+m112+m113;
```

```
    m121=polyint(conv(b1,B(13,:)),x(1),x(2));
    m122=polyint(conv(b2,B(14,:)),x(2),x(3));
    m123=polyint(conv(b3,B(15,:)),x(3),x(4));   m12=m121+m122+m123;

    m131=polyint(conv(b2,B(17,:)),x(2),x(3));
    m132=polyint(conv(b3,B(18,:)),x(3),x(4));   m13=m131+m132;

    m14=polyint(conv(b3,B(21,:)),x(3),x(4));

    m221=polyint(conv(B(13,:),B(13,:)),x(1),x(2));
    m222=polyint(conv(B(14,:),B(14,:)),x(2),x(3));
    m223=polyint(conv(B(15,:),B(15,:)),x(3),x(4));
    m224=polyint(conv(B(16,:),B(16,:)),x(4),x(5));
    m22=m221+m222+m223+m224;

    m231=polyint(conv(B(14,:),B(17,:)),x(2),x(3));
    m232=polyint(conv(B(15,:),B(18,:)),x(3),x(4));
    m233=polyint(conv(B(16,:),B(19,:)),x(4),x(5));   m23=m231+m232+m233;

    m241=polyint(conv(B(15,:),B(21,:)),x(3),x(4));
    m242=polyint(conv(B(16,:),B(22,:)),x(4),x(5));   m24=m241+m242;

    m25=polyint(conv(B(16,:),B(25,:)),x(4),x(5));

    %  Build M22 matrix:
      w1=[m11 m22*ones(1,n-3) m11]; w2=[m12 m23*ones(1,n-4) m12];
      w3=[m13 m24*ones(1,n-5) m13]; w4=[m14 m25*ones(1,n-6) m14];
      M221=diag(w1) + diag(w2,1) + diag(w3,2) + diag(w4,3);
      M222=diag(w2,-1) + diag(w3,-2) + diag(w4,-3);
      M22=M221+M222;    M22=rhob*M22;
end
end   %  End matm22.m
***************************************************************************
```

5. Function file *a3132.m* computes A_{31}^N and A_{32}^N.

```
*****************************************************************
function [A31,A32]=a3132(a,b,n,mx,my,rhof)

%  [A31 A32] = a3132(a,b,n,mx,my,rhof)
%
%  Currently, this function returns matrices A31 and A32.  The elements
%  of these matrices correspond to the integrated product of the
%  collapsed tensored Legendre poly's (i.e., y=0) and the cubic poly's
%  over [a,b]. Note: The cubic poly's satisfy clamped beam boundary
%  conditions. A32 is formed by computing -A31'.  Integration is
%  accomplished using Gaussian quadrature.
%
%  Input:  [a,b] = interval of integration (i.e., length of beam)
%          n = number of symmetric partitions [a,b] is divided into
%          mx = highest degree of Legendre poly in basis set for beam
%          my = "       "       "       "       "       " for cavity
%          rhof = density of fluid
%  Extrensic functions called:  legtrans.m to compute Legendre poly's
%                               myspline.m to compute cubic splines
%  Written by Major J. M. Shehan, 13 May 95.

%  Begin a3132.m

% Compute Gaussian quadrature weights and knots for partitioned beam.
   h=b/n;  v=0:h:b;

   k=round((4+mx)/2);    % Determine no. of knots.
   x=1:1:k-1;  x=x./sqrt((2*x+1).*(2*x-1));  j=diag(x,1)+diag(x,-1);
   [u x]=eig(j);  x=diag(x);  [x i]=sort(x);  u=u(:,i);  w=u(1,:).^2;
   w=w.*2;  % 'w' denotes weights; 'x' denotes knots.
% Translate knots to appropriate interval.
   dx=h/2;  u=1:2:2*n;  x=x';
   for i=1:n
      X(i,:)=dx*(x+u(i));
   end
% Compute Legendre & cubic poly's.
   [L]=legtrans(b,mx);  [B,B1,Bnm1]=myspline(a,b,n);
% Evaluate Legendre's at translated knots corresp to beam partitions.
      P(1:n,1:k)=ones(n,k);  s=0;
```

```
       for i=2:mx+1
         for j=1:n
            P(j+n+s,:)=polyval(L(i,:),X(j,:));
         end;    s=s+n;
       end
% Compute integrals involving B(1) and B(n-1) cubic splines.
     M=zeros(n-1,mx+1);   % Allocate storage space.
     q1=w.*polyval(B1(1,:),X(1,:));   q2=w.*polyval(B1(2,:),X(2,:));
     q3=w.*polyval(B1(3,:),X(3,:));
     r1=w.*polyval(Bnm1(1,:),X(n-2,:));
     r2=w.*polyval(Bnm1(2,:),X(n-1,:));
     r3=w.*polyval(Bnm1(3,:),X(n,:));
     s=0;
     for i=1:mx+1
       v1=sum(q1.*P(1+s,:));   v2=sum(q2.*P(2+s,:));
       v3=sum(q3.*P(3+s,:));
       u1=sum(r1.*P(n-2+s,:));   u2=sum(r2.*P(n-1+s,:));
       u3=sum(r3.*P(n+s,:));
       M(1,i)=dx*(v1+v2+v3);   M(n-1,i)=dx*(u1+u2+u3);   s=s+n;
     end
% Compute interior integrals (B(i),P(j)) for i=2,...,n-2 & for j=1:mx+1.
     [uu vv]=size(B);
     s=0;   r=0;
     s1=w.*polyval(B(13,:),X(1,:));   s2=w.*polyval(B(14,:),X(2,:));
     s3=w.*polyval(B(15,:),X(3,:));   s4=w.*polyval(B(16,:),X(4,:));
     for i=1:(uu-24)/4
         for j=1:mx+1
            t1=sum(s1.*P(1+s+r,:));   t2=sum(s2.*P(2+s+r,:));
            t3=sum(s3.*P(3+s+r,:));   t4=sum(s4.*P(4+s+r,:));
            M(i+1,j)=dx*(t1+t2+t3+t4);   s=s+n;
         end;   s=0;   r=1+r;
     end;   M=M';
% Generate A31 & A32 using M and fact that P(y=0)=(-1) or (1) for all
%collapsed Legendre poly's.
     m=(mx+1)*(my+1)-1;   A31=zeros(m+1,n-1);   s=0;
     for i=1:(my+1)
         A31(1+s:mx+1+s,:)=(-1)^(i+1)*M;      s=s+mx+1;
     end
     A31=-rhof*A31(2:m+1,:);   A32=-A31';


end   % End a3132.m
*****************************************************************
```

6. Function file *mata41.m* computes A_{41}^N.


```
***********************************************************************
function [A41]=mata41(b,el,mx,my,rhof)

% [A41]=mata41(b,el,mx,my,rhof)
%
% Computes matrix A41 for Model II.
% Input variables: b = right bdary of [0,b].
%                  el = upper bdary of [0,el],
%                  mx = highest degree of Legendre poly for beam,
%                  my = highest degree of Legendre poly for cavity,
%                  cc = proportionality constant of damping term.
% In this program, matrix A41A corresponds to integration over
% 0<=y<=el, x=0; A41B corresponds to integration over 0<=x<=b, y=el;
% and A41C corresponds to integration over 0<=y<=el, x=b.
%
% Written by Major J. M. Shehan; last update: 23 May 95.


mx1=mx+1; my1=my+1; m=mx1*my1-1; m1=m+1;   % Notation simplification.


%%% This algorithm can be used to computes A41C.
%   y=ones(1,my1); v=1:2:2*my1;  intPP=el*(y./v); T=ones(mx1,my1);
%   A41C=zeros(m1,m1); q=0; g=0;
%   for i=1:my1
%       A41C(1+q:mx1+q,1+g:my1+g)=intPP(i)*T;   q=mx1+q; g=my1+g;
%   end;   A41C=2*A41C(2:m1,2:m1);


%%%  This algorithm computes A41A & A41C simultaneously.
y=ones(1,my1); vy=1:2:2*my1; intPPy=el*2*(y./vy);  A41AC=zeros(m1,m1);
  s=0; q=mx1; g=my1;
  for i=1:mx1
    for j=1:my1
      Block(i,j)=1+(-1)^(j+1+s);
    end;  s=s+1;
  end
  for i=2:my1
       A41AC(1+q:mx1+q,1+g:my1+g)=intPPy(i)*Block;   q=mx1+q; g=my1+g;
  end;   A41AC=A41AC(2:m1,2:m1);
A41AC(1:mx,1:my)=intPPy(1)*Block(1:mx,2:my1);
```

```
%%%  This alogrithm computes A41B.
x=ones(1,mx1); vx=1:2:2*mx1; intPPx=b*(x./vx);  t1=intPPx(1);
t2=intPPx(mx1);  intPPx(1)=t2;  intPPx(mx1)=t1; D=diag(intPPx);
A41B=zeros(m1,m1);   g=0;
  for i=1:my1
      A(1:mx1,1+g:mx1+g)=D;  g=g+mx1;
    end
  for k=1:my
    A41B(k*(mx1)+1:(k+1)*(mx1),:)=A;
  end;  A41B(1:mx1,:)=A; A41B=A41B(1:m,1:m);

%%% Compute A41
A41=-rhof*(sparse(A41AC)+sparse(A41B));

end  % End mata41.m
*************************************************************************
```

7. Function file *myspline.m* computes the cubic splines.


```
****************************************************************************
function [B,B1,Bnm1] = myspline(a,b,n)

%  [B,B1,Bnm1] = myspline(a,b,n)
%
%  This function returns a family of standard cubic splines
%  defined over [a,b], as well as the "boundary splines" which
%  satisfy clamped boundary conditions. The interval [a,b] is
%  divided into 'n' uniform partitions; 'h' is the step size.
%
%  Input:  [a,b] = interval along x-axis;
%             n = no. of equispaced partitions of [a,b].
%  Output: B = each row of "matrix" B corresponds to a cubic poly
%              which is defined only over one step size (b-a)/n;
%              every 4 rows constitute a piecewise smooth cubic
%              polynomial which is non-zero only over 4 intervals
%              (e.g., rows 1-4 is the first cubic poly, rows 5-8
%              makes up the second basis function, etc.).
%          B1 = left most cubic spline satisfying clamped boundary
%              conditions.
%          Bnm1 = right most cubic spline satisfying clamped boundary
%               conditions.
%  Written by Major J. M. Shehan, updated 10 April 95.

%  Begin myspline.m

%  Form standard cubic splines.

h=(b-a)/n;    x=a-3*h:h:b+3*h;    hh=1/h^3;    z=0;

for i = 1:n+3
B(i+z,:)=hh*[1    -3*x(i)    3*x(i)^2    -x(i)^3];
B(i+1+z,:)=hh*[-3 3*h+9*x(i+1) 3*h^2-6*h*x(i+1)-9*x(i+1)^2
                      h^3-3*h^2*x(i+1)+3*h*x(i+1)^2+3*x(i+1)^3];
B(i+2+z,:)=hh*[3 3*h-9*x(i+3) 9*x(i+3)^2-6*h*x(i+3)-3*h^2
                      h^3+3*h^2*x(i+3)+3*h*x(i+3)^2-3*x(i+3)^3];
B(i+3+z,:)=hh*[-1    3*x(i+4)    -3*x(i+4)^2    x(i+4)^3];
z=z+3;
end
```

```
% Form exterior splines which satisfy clamped boundary conditions.
    %  Form B1=B(0)-2*B(1)-2*B(-1).
         b1=B(7,:)-2*B(10,:)-2*B(4,:);  b2=B(8,:)-2*B(11,:);
         b3=-2*B(12,:);
         B1=[b1; b2; b3];
    % Form Bnm1 = B(n) - 2*B(n+1) - 2*B(n-1).
         [uu vv]=size(B);
         bnm11=-2*B(uu-11,:); bnm12=B(uu-7,:)-2*B(uu-10,:);
         bnm13=B(uu-6,:)-2*B(uu-9,:)-2*B(uu-3,:);
         Bnm1=[bnm11; bnm12; bnm13];
end  %  End myspline.m
****************************************************************
```

8. Function file *legtrans.m* computes the transformed Legendre polynomials.

```
*********************************************************************
function [L] = legtrans(b,n)

%   [L] = legtrans(b,n)
%
%   This function produces a 'matrix' L whose rows are translated
%   Legendre  polynomials of 0th through nth degree defined on [0,1].
%   The 0th   degree  polynomial corresponds to the first row of the
%   output matrix, while the nth degree polynomial corresponds to the
%   n+1 row (i.e., the last row) of the output matrix.
%
%   Input arguments:  n = highest degree of translated Legender poly
%                         desired;
%       b = right end point of interval assuming [0,b].
%
%   Algorithm written by Major J. M. Shehan, updated 11 April 95.

%   Begin legtrans.m

%   Generate tranlated Legendre poly's of 0th & 1st degree.
        L(1,:)=[zeros(1,n) 1]; L(2,:)=[zeros(1,n-1) 2/b -1];

%   Generate 2d-nth deg translated Legendre poly's recursively.
k=0; r=n+1;
for i=2:n
  d=i+1; p=[(2*(i-1)+1)*2/b (2*(i-1)+1)*(-1)];
  L(d,:)=(1/i)*([zeros(1,n-i) conv(p,L(i,n-k:r))] - [(i-1)*L(i-1,:)]);    .
  k=k+1;
end
end    % End legtrans.m
*********************************************************************
```

72

9. Function file *polyint.m* performs polynomial integration.


```
********************************************************************
function [polyint] = polyint(p,a,b)

%  This function integrates the polynomial 'p' over the interval [a,b].
%  The polynomial 'p' is written as a vector 'v' with coefficients listed
%  in descending order (e.g., 3x^2 + 5x - 8 ===>  [3 5 8]).
%
%  Written by Major J. M. Shehan 10 Feb 95.

%  Begin polyint.m

v=[p 0]; y=v./[length(p):-1:1 1]; polyint = polyval(y,b) - polyval(y,a);

end    % End polyint.m
********************************************************************
```

REFERENCES

[1] H. T. Banks, K. Ito and C. Wang, *Exponentially Stable Approximations Of Weakly Damped Wave Equations.* Center for Applied Mathematical Sciences (CAMS 91-12), University of Southern California, Los Angeles, California, 1991.

[2] M. C. Delfour and M. P. Polis, *On Issues Related to Stabilization of Large Flexible Structures.* Frontiers In Applied Mathematics, Control and Estimation in Distributed Parameter Systems, Society for Industrial and Applied Mathematics, Vol. 11, 1992.

[3] G. Chen, S. A. Fulling, F. J. Narcowich, and S. Sun, *Exponential Decay Of Energy Of Evolution Equations With Locally Distributed Damping.* Society for Industrial and Applied Math, Vol. 51, No. 1, February 1991.

[4] M. Vidyasagar, *Nonlinear Systems Analysis.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.

[5] Fariba Fahroo and Chunming Wang, *Stabilization of Undamped Or Weakly Damped Wave Equations Via Coupling With Flexible Beams.* To appear.

[6] H. T. Banks, W. Fang and K. Ito, *Exponential Stability of Second Order Coupled Systems By Energy Multiplier Method.* To appear.

[7] H. T. Banks and Yun Wang, *Distributed Parameter System Models For Damage Detection And Location In Smart Material Structures.* Proceedings, Vol 2192, SPIE-The International Society for Optical Engineering, Bellingham, Washington.

[8] Kôsaku Yosida, *Functional Analysis.* Springer-Verlag, Berlin Heidelberg, 1978.

[9] H. T. Banks, W. Fang, R. J. Silcox and R. C. Smith, *Approximation Methods for Control of Acoustic/Structure Models with Piezoceramic Actuators.* National Aeronautics and Space Administration Cr-189578, Institute for Computer Applications in Science and Engineering Report No. 91-88, Langley Research Center, Hampton, Virginia, 1991.

[10] Ferenc Szidarovszky and Sidney Yakowitz, *Principles and Procedures of Numerical Analysis.* Plenum Press, New York, 1978.

[11] P. M. Prenter, *Splines and Variational Methods.* John Wiley & Sons, Inc., 1975.

[12] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis.* SpringerVerlag, New York, 1993.

[13] Carl de Boor, *A Practical Guide to Splines.* SpringerVerlag, New York, 1978.

[14] Gene H. Golub and Charles F. Van Loan, *Matrix Computations.* The Johns Hopkins University Press, Baltimore, Maryland, 1983.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center   2
   Cameron Station
   Alexandria, VA 22304-6145

2. Library, Code 52   2
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Chairman, Code MA   1
   Department of Mathematics
   Naval Postgraduate School
   Monterey, CA 93943

4. Professor Fariba Fakhroo, Code MA/FF   5
   Department of Mathematics
   Naval Postgraduate School
   Monterey, Ca 93943

5. Professor Van Emden Henson, Code MA/HV   1
   Department of Mathematics
   Naval Postgraduate School
   Monterey, CA 93943

6. Director, Training and Education   1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, VA 22134-5027

7. Major Joe M. Shehan   2
   Department of Mathematics
   United States Naval Academy
   Annapolis, MD 21402